

# Development of Secure IPsec Tunnelling in a Mobile IP Architecture

---

Vincent Pau

Supervisor: Assoc. Prof. Ray Hunt

10 November 2005



## **Abstract**

Internet Protocol security (IPsec) is a widely accepted standard for securing IP network traffic but has limited functionality in a Mobile IP environment. The aim of this research is to develop a solution that enables mobile nodes to handoff IPsec tunnels in a transparent manner when moving between different networks. Previous researches suggest two general approaches to solving this problem: to run IPsec over Mobile IP, or to dynamically update the IPsec tunnel endpoints. As part of this research, we proposed a variation of the latter approach, whereby Mobile IP registration messages are used to update the IPsec tunnel endpoints. The solution enables a mobile node to establish an IPsec tunnel once and maintain the tunnel across handoffs. A testbed was developed for evaluating the performance of the various approaches under different handoff conditions. The proposed solution was implemented and tested successfully on the testbed, proving its feasibility. The study also compares the performance of the proposed solution against running IPsec over Mobile IP, and the current approach of re-establishing new IPsec tunnels. Although the proposed solution is more complex compared to running IPsec over Mobile IP, the results show that it is more efficient in terms of bandwidth overhead. The results also show that the proposed solution has a lower handoff delay compared to the current approach of re-establishing new IPsec tunnels.

# Acknowledgements

I would like to take this opportunity to express my gratitude to all those who have made the completion of this report possible. First of all, special thanks are due to my supervisor, Ray Hunt, as well as Mayank Keshariya, for their valuable guidance, suggestions, insights and support throughout the course of this study. Gratitude must also be extended to Anita Pau for proof-reading this report. Last, but not least, I would like to thank my family and friends for their support and understanding throughout the year.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Background Information</b>	<b>5</b>
2.1	Internet Protocol Security . . . . .	5
2.2	Mobile IP . . . . .	7
<b>3</b>	<b>Research Aim</b>	<b>9</b>
<b>4</b>	<b>Solutions for Achieving IPsec Mobility</b>	<b>10</b>
4.1	IPsec over Mobile IP . . . . .	10
4.2	Dynamic Update of IPsec Tunnel Endpoints . . . . .	11
4.2.1	Additional ISAKMP Information Exchange Messages . . . . .	12
4.2.2	Mobile IP Registration Messages . . . . .	12
<b>5</b>	<b>Testbed Implementation</b>	<b>15</b>
5.1	Network Configuration . . . . .	15
5.2	Solution Implementation . . . . .	16
5.2.1	Disabled Path MTU Discovery . . . . .	18
5.2.2	Forced Handoff . . . . .	19
5.2.3	Use of Home Address as the Mobile Node's Source IP Address . . . .	19
5.2.4	Split Tunnelling for the Address Change Notification System . . . .	20
5.2.5	Firewall Rules for Blocking Insecure Communication during Handoff	20
5.2.6	Updating the SAD and SPD . . . . .	21
<b>6</b>	<b>Performance Study</b>	<b>22</b>
6.1	Bandwidth Overhead . . . . .	22
6.1.1	Ping Latency . . . . .	22
6.1.2	MN-HA Bandwidth Overhead . . . . .	23
6.1.3	HA-MN Bandwidth Overhead . . . . .	24
6.1.4	FTP Bandwidth and Transfer Time . . . . .	25
6.2	Handoff Latency . . . . .	27
6.2.1	Pre-shared Key vs. RSA Signature Authentication . . . . .	29
6.3	Maintaining FTP File Transfer During Handoff . . . . .	30

<b>7</b>	<b>Security Implications</b>	<b>32</b>
<b>8</b>	<b>Areas for Future Research</b>	<b>33</b>
<b>9</b>	<b>Conclusions</b>	<b>34</b>
<b>A</b>	<b>Wireless Access Point Configuration</b>	<b>35</b>
A.1	IPSEC-HN Configuration (IPSEC-HN.cnf) . . . . .	35
A.2	IPSEC-FN1 Configuration (IPSEC-FN1.cnf) . . . . .	36
<b>B</b>	<b>Dynamics Mobile IP Configuration</b>	<b>37</b>
B.1	Dynamics Home Agent Daemon Configuration (dynhad.conf) . . . . .	37
B.2	Dynamics Mobile Node Daemon Configuration (dynmnd.conf) . . . . .	39
<b>C</b>	<b>Racoon Configuration</b>	<b>42</b>
C.1	Home Agent Racoon Configuration . . . . .	42
C.2	Mobile Node Racoon Configuration . . . . .	43
<b>D</b>	<b>Basic Network Configuration Scripts</b>	<b>44</b>
D.1	Home Agent Setup Script (ha-setup) . . . . .	44
D.2	Foreign Router Setup Script (fa-setup) . . . . .	45
D.3	Mobile Node Setup Script (mn-setup) . . . . .	45
<b>E</b>	<b>Standard IPsec Scripts</b>	<b>47</b>
E.1	Home Agent IPsec Script (ha-ipsec) . . . . .	47
E.2	Mobile Node IPsec Setup Script (mn-ipsec-setup) . . . . .	47
E.3	Mobile Node IPsec Handoff Script (mn-ipsec-handoff) . . . . .	48
E.4	Mobile Node IPsec Teardown Script (mn-ipsec-teardown) . . . . .	49
<b>F</b>	<b>MIPsec Scripts</b>	<b>50</b>
F.1	Home Agent MIPsec Setup Script (ha-mipsec) . . . . .	50
F.2	Mobile Node MIPsec Setup Script (mn-mipsec-setup) . . . . .	52
F.3	Mobile Node MIPsec Handoff Script (mn-mipsec-handoff) . . . . .	53
F.4	Mobile Node MIPsec Teardown Script (mn-mipsec-teardown) . . . . .	54
<b>G</b>	<b>DynIPsec Scripts</b>	<b>56</b>
G.1	Home Agent DynIPsec Script (ha-dynipsec) . . . . .	56
G.2	Mobile Node DynIPsec Setup Script (mn-dynipsec-setup) . . . . .	58
G.3	Mobile Node DynIPsec Handoff Script (mn-dynipsec-handoff) . . . . .	59
G.4	Mobile Node DynIPsec Teardown Script (mn-dynipsec-teardown) . . . . .	60
<b>H</b>	<b>Update Address</b>	<b>62</b>
H.1	Source Code (updaddr.c) . . . . .	62

# Chapter 1

## Introduction

Developments in wireless network technologies have enabled users to become more mobile. Security is a major concern, especially for mobile corporate users who often require secure access to the corporate network. Internet Protocol security (IPsec) is a widely accepted standard for securing network traffic. It was developed to provide various security services for Internet Protocol (IP) traffic in both IP version 4 (IPv4) and IP version 6 (IPv6) networks. The set of services includes access control, data origin authentication, data integrity, and data confidentiality. Upper layer security protocols, such as Secure Socket Layer (SSL), only provide security for a subset of the services running over IP. On the other hand, link layer security protocols, such as Wired Equivalent Privacy (WEP) and WIFI Protected Access (WPA), only work on a particular data link protocol. Numerous security flaws have been discovered in WEP and WPA. This makes IPsec an ideal security solution, as it protects all traffic running over IP while maintaining interoperability between all systems using IP. Many Virtual Private Networks (VPNs) are utilising IPsec to provide mobile users with access to corporate resources. However, IPsec does have its limitations when operating in a mobile environment.

Section 2 provides some background information on IPsec and Mobile IP. Section 3 outlines the goals of this research. Section 4 describes the various approaches for handling IPsec tunnels in a Mobile IP architecture, including our proposed solution. The testbed implementation is detailed in Section 5. An analysis of the performance of the various approaches is set out in Section 6. Section 7 discusses the security implications of the various solutions. Section 8 identifies some areas for future research, followed by Section 9, which presents the conclusions of this research.

# Chapter 2

## Background Information

### 2.1 Internet Protocol Security

The IPsec protocol, as defined in RFC 2401 [12], is designed to provide interoperable, high quality, cryptographically-based security for IP networks. Traffic security is provided using either the Authentication Header (AH) protocol, or the Encapsulating Security Payload (ESP) protocol, which combines the functions of authentication and encryption. Both AH and ESP support two modes of use: transport mode and tunnel mode. Transport mode provides protection primarily for the payload of an IP packet, while tunnel mode protects the entire IP packet. With tunnel mode, the original IP packet is encapsulated within a new IP packet. Depending on the tunnel configuration, the IP source and destination addresses of the new IP header may be different from that of the original IP header. IPsec can also be configured to use different authentication and encryption algorithms, according to the needs of the users, making it a very robust and flexible solution.

Security Association (SA) is a key concept in the IPsec architecture. An SA is a one-way relationship between a sender and a receiver that affords security services to the traffic it carries. The information that is necessary for processing IPsec traffic is stored in the Security Policy Database (SPD) and the Security Association Database (SAD). The SPD specifies the policies that are to be applied to the traffic destined to or originated from a given host or network, while the SAD maintains the parameters associated with all active SAs. An SA is uniquely identified by three parameters: a Service Parameter Index (SPI), an IP destination address, and a security protocol (AH or ESP) identifier. Figure 2.1 shows a sample SAD entry. Each SA can afford exactly one security protocol, either AH or ESP, but not both. Sometimes a security policy may call for a combination of services for a particular traffic flow that is not achievable with a single SA. In such instances, it will be necessary to employ multiple SAs to implement the required security policy.

Both IPsec and the underlying IP protocol were originally designed for fixed networks and therefore have limitations in a mobile environment. In order to maintain IP layer functionality when moving between different networks, a mobile node must obtain a new IP address that reflects its most current point of attachment. This change in IP address



20.0.0.1 30.0.0.11 The IP destination address will become invalid when the mobile node changes its IP address.

```

esp mode=tunnel spi=769(0x00000301) reqid=0(0x00000000)
E: 3des-cbc c9e7914b 4d5ce2b8 bbaee2ee lcaa51fb 9elffbbc 2ab40b0c
A: hmac-md5 c47cdfd2 1b5d5246 c475cede 4e385b92
seq=0x00000000 replay=0 flags=0x00000000 state=mature
created: Jun 30 14:28:41 2005 current: Jun 30 14:38:08 2005
diff: 567(s) hard: 0(s) soft: 0(s)
last: Jun 30 14:31:14 2005 hard: 0(s) soft: 0(s)
current: 4624(bytes) hard: 0(bytes) soft: 0(bytes)
allocated: 34 hard: 0 soft: 0
sadb_seq=0 pid=18484 refcnt=0

```

Figure 2.1: A sample SAD entry.

may cause certain applications and services running over the IP layer to fail, including IPsec. When a mobile node changes its IP address, the IP address parameters in the SPDs and SADs become invalid. As a result, IPsec is no longer able to identify the SAs that were previously associated with it.

Currently, a mobile node has to break off and re-establish its IPsec tunnels when moving between different networks. This disrupts and breaks applications and services running over the IPsec tunnels. Setting up an IPsec tunnel is a computation-intensive process which involves key generations and SA negotiations. This process significantly increases the handoff latency, especially in a heavily loaded network, and puts additional processing load on the end nodes.

The SA negotiation process consists of two phases: the *Internet Security Association and Key Management Protocol*<sup>1</sup> (ISAKMP) SA negotiation and the IPsec SA negotiation. The purpose of the phase 1 ISAKMP SA negotiation is to validate the identity of the remote peer and to protect the phase 2 IPsec SA negotiation. There are three ways to authenticate the remote peer: pre-shared keys, *DSA*<sup>2</sup> digital signatures, or *RSA*<sup>3</sup> signatures. In practice, the two commonly used methods are pre-shared keys and RSA signatures. Pre-shared key uses a shared-key that has been previously set up on both the client and the server. In contrast, RSA signature uses a public key infrastructure (PKI) to authenticate the remote

<sup>1</sup>Internet Security Association and Key Management Protocol (ISAKMP) is a protocol for establishing SAs and cryptographic keys in an Internet environment. ISAKMP defines the procedures for authenticating a communicating peer, creation and management of SAs, key generation techniques, and threat mitigation. ISAKMP typically utilises IKE for key exchange, although other methods can be implemented. ISAKMP is documented in RFC 2408 [13].

<sup>2</sup>The Digital Signature Algorithm (DSA) is a United States Federal Government standard for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) for use in their Digital Signature Standard (DSS).

<sup>3</sup>RSA is an Internet encryption and authentication system that uses an algorithm developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman.

peer. A single phase 1 ISAKMP SA can be used to establish multiple phase 2 IPsec SAs. In the phase 2 IPsec SA negotiation, all the parameters of the IPsec tunnel are negotiated.

## 2.2 Mobile IP

There has been significant research and development in the area of IP mobility. Request for Comments (RFC) 3344 [16] and 3775 [10] define standards for IP mobility support in IPv4 and IPv6 networks, respectively. Mobile IP addresses the issue of a mobile node's change in point of attachment. With Mobile IP, a mobile node has two IP addresses: a home address which is permanently associated with the mobile node and a "care-of-address" (CoA) which reflects the mobile node's most current point of attachment. Whenever the CoA changes, the mobile node registers its new CoA with its home agent. A home agent is simply a router on the mobile node's home network which is capable of supporting Mobile IP. The Mobile IP registration is a simple two step process. Firstly, the mobile node sends a Mobile IP registration request containing its current CoA to the home agent. The home agent then sends a Mobile IP registration reply to the mobile node indicating whether the registration has been successful.

There are two types of CoA: a foreign agent CoA and a co-located CoA. A foreign agent CoA is an address of a foreign agent with which the mobile node is registered, while a co-located CoA is an externally obtained local address which the mobile node has associated with one of its own network interfaces [16]. A foreign agent is a router on a mobile node's visited network which provides routing services to the mobile node while registered. Using a co-located CoA has the advantage of allowing a mobile node to function without a foreign agent.

When the mobile node is in the home network, all IP packets to and from the mobile node are routed using standard IP routing. However, if an IP packet is sent to the mobile node's home address while it is away from home, the home agent intercepts the IP packet and tunnels the traffic to the mobile node's CoA, as shown in Figure 2.2. If a foreign agent CoA is used, the foreign agent detunnels the packet before forwarding it to the mobile node. Otherwise, the decapsulation is performed by the mobile node itself. This approach ensures that a mobile node is always reachable using its home address. IP packets sent by the mobile node can be delivered to its destination using either triangular routing or reverse tunnelling. With triangular routing, the IP packets are delivered normally using standard IP routing. However, since IP packets sent by the mobile node uses its home address as the source address, these packets may be dropped by the routers in the foreign network. Many routers implement ingress filtering that do not allow forwarding of packets that have a source address which appears topologically incorrect. In such environments, the mobile node may use reverse tunnelling. With reverse tunnelling, the packets from the mobile node are tunnelled to the home agent. The home agent then decapsulates the packets before delivering them using standard IP routing.

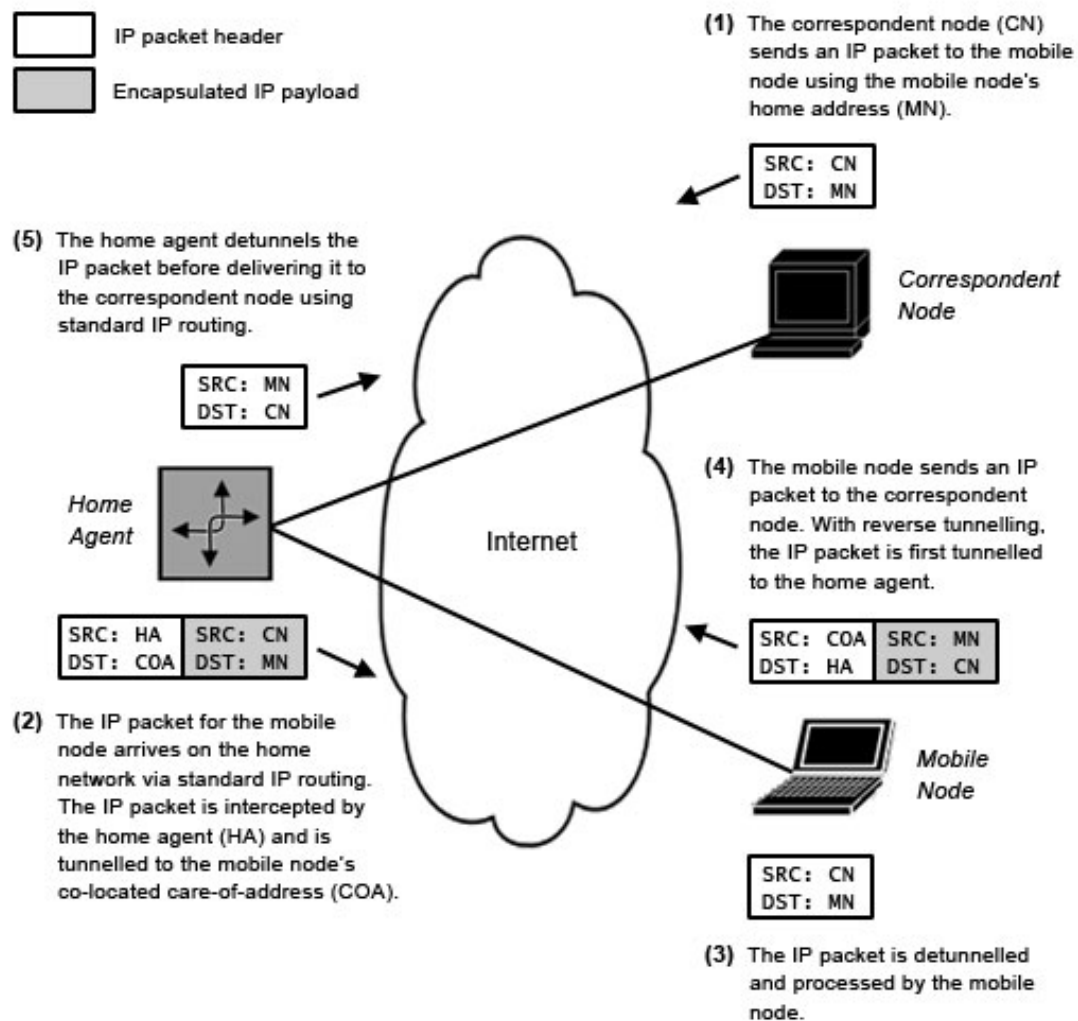


Figure 2.2: Mobile IP routing process when operating in co-located CoA mode with reverse tunnelling.

# Chapter 3

## Research Aim

The aim of this research is to develop a solution that enables mobile nodes to handoff IPsec tunnels in a transparent manner when moving between different IPv4 networks. We propose a solution, which is further described in Section 4.2.1, whereby Mobile IP registration messages are used to dynamically update the IPsec tunnel endpoints. The solution will enable a mobile node to establish an IPsec tunnel once and maintain the tunnel across handoffs. This eliminates the extra processing required to setup new IPsec tunnels. As a result, it should reduce the handoff latency. Additionally, all connection-oriented services and applications running over the IPsec tunnel should also be maintained. This makes IPsec a more practical solution in a mobile environment.

Both IPsec and Mobile IP support various options and modes of operation. The focus of this research is on maintaining an IPsec tunnel operating in tunnel mode between a mobile node and its home agent, as in the case where a mobile user is trying to access the corporate network. We are only considering the case where no foreign agents are involved, as it is unreasonable to assume that an organisation has control over the management and configuration of the mobile node's visited networks. Therefore, Mobile IP is configured to operate in co-located CoA mode and reverse tunnelling is employed, to support ingress filtering, for packets sent from the mobile node.

## Chapter 4

# Solutions for Achieving IPsec Mobility

There are several researches which address the issue of IPsec mobility. These researches suggest two general solutions to the problem: to run IPsec over Mobile IP or to dynamically update the IPsec tunnel endpoints.

### 4.1 IPsec over Mobile IP

The simpler of the two solutions is to run IPsec over Mobile IP. First, the mobile node establishes a Mobile IP tunnel with the home agent using the mobile node's CoA. Then, an IPsec tunnel is established using the mobile node's permanent home address to run over the Mobile IP tunnel. Figure 4.1 shows how the IP packet is processed when running IPsec over Mobile IP. This solution enables the IPsec tunnel to be maintained while moving between different networks. However, it is inefficient as the double tunnelling involved introduces overhead due to the extra IP header that is added to each IP packet. The overhead is significant, especially when the path *maximum transmission unit*<sup>1</sup> (MTU) is small; for example, when running over older devices on the Internet, or a third generation (3G) wireless wide area network (WAN).

The Portland State University (PSU) Secure Mobile Networking Project [17] has managed to integrate KAME IPsec, an open source IPsec implementation, into its Mobile IP implementation. Its implementation, based on running IPsec over Mobile IP, enables a mobile node to have a two-way ESP tunnel between itself and its partner home agent. In theory, this approach should work for any type of IPsec configuration. Currently, the PSU Mobile IP implementation only works on FreeBSD systems, with limited support available for Linux systems.

---

<sup>1</sup>The maximum transmission unit (MTU) is the largest size packet or frame, specified in octets (eight-bit bytes), that can be sent in a packet-based or frame-based network such as the Internet.

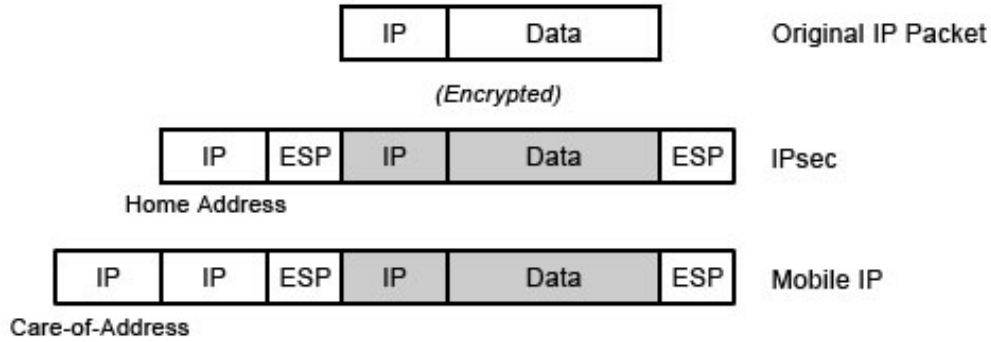


Figure 4.1: The IP packet structure when running IPsec over Mobile IP.

## 4.2 Dynamic Update of IPsec Tunnel Endpoints

The alternative solution is to dynamically update the SAD and SPD entries in order to maintain the IPsec tunnels. This solution was originally suggested by Byoung-Jo and Srinivasan [4]. In this case, the IPsec tunnel is established using the CoA, as shown in Figure 4.2, eliminating the need for double tunnelling. Whenever the mobile node changes its CoA, the correspondent node must be notified so that the database entries can be updated to reflect the new CoA. This ensures that future packets are sent to the new address. The address change notification system should be secure against attacks to prevent an attacker from diverting the IPsec tunnel.

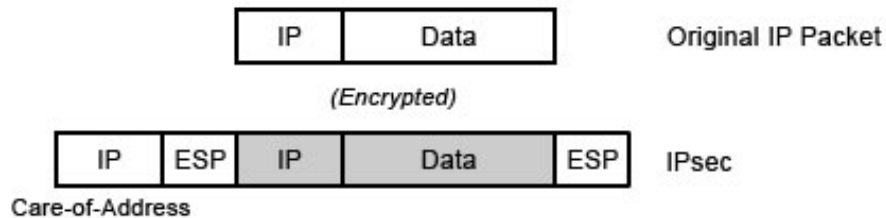


Figure 4.2: The IP packet structure with dynamic update of IPsec tunnel endpoints.

This approach only works for IPsec tunnels operating in tunnel mode as it involves the manipulation of the new IP header added by IPsec. When operating in transport mode, an extra Mobile IP encapsulation is required to allow the packets to be routed to and from the mobile node.

### 4.2.1 Additional ISAKMP Information Exchange Messages

Byoung-Jo and Srinivasan [4] suggest the use of additional ISAKMP information exchange messages for the address change notification system. The suggested solution defines two new message types: one to notify of a tunnel address update, and the other to confirm the success of the address update. Figure 4.3 shows an example of how this solution would work in practice. Using this solution, the newly defined messages are protected using IPsec and should therefore be secure against attack. However, without any modifications, these messages cannot be decrypted by the mobile node as the IPsec tunnels are no longer valid after the handoff. To overcome this problem, the solution removes the dependence on IP destination address for uniquely identifying an SA. When processing inbound unicast IPsec tunnels at a node, all the IPsec destination addresses will be the same. In this situation, an SA will be identified primarily using the SPI. The IP destination address is only necessary when handling IPsec multicasts and broadcasts, which are rarely used. According to RFC 2401 [12], the destination system will normally select the SPI value in the case of unicast traffic. Thus, it is trivial for the destination node to ensure that a chosen SPI is unique system-wide. Although the dependence on IP destination address has been removed, the correspondent node must still update the IP destination address parameters in the SAD for the corresponding SA when the mobile node changes its CoA. This solution has been proven to work but no publicly available implementation of this solution is currently available.

### 4.2.2 Mobile IP Registration Messages

As part of this research, we propose the use of an alternative address change notification system. Instead of defining additional ISAKMP information exchange messages, we propose the use of Mobile IP registration messages for informing the remote peer of any changes to the mobile node's CoA, as shown in Figure 4.4. Furthermore, our solution does not remove the dependence on IP destination address for uniquely identifying an SA and relies on the Mobile IP authentication mechanism for securing the registration messages. This allows for a potentially simpler implementation as it simply requires the integration of existing IPsec and Mobile IP implementations. In practice, Mobile IP registration messages are also used to renew a Mobile IP registration. However, it is trivial to differentiate between these two cases by keeping track of the current CoA, and trigger an update of the SAD and SPD when necessary. Additionally, our solution still works for multicast and broadcast addresses. The use of Mobile IP registration messages does, however, require the IPsec tunnel to be terminated at the home agent.

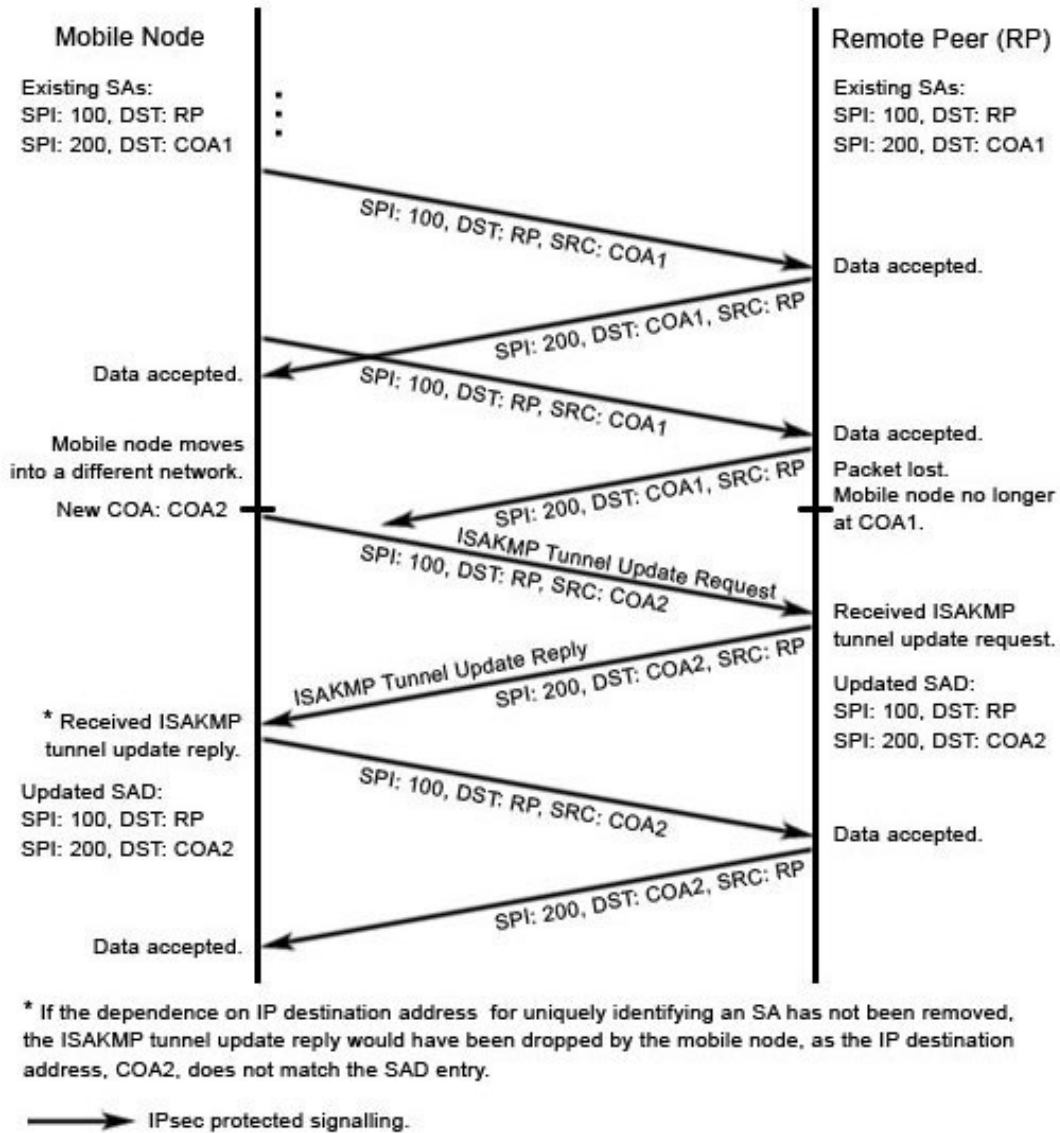


Figure 4.3: The IPsec tunnel handoff process when using additional ISAKMP information exchange messages, as suggested by Byoung-Jo and Srinivasan [4], for dynamic updates of the IPsec tunnel endpoints.



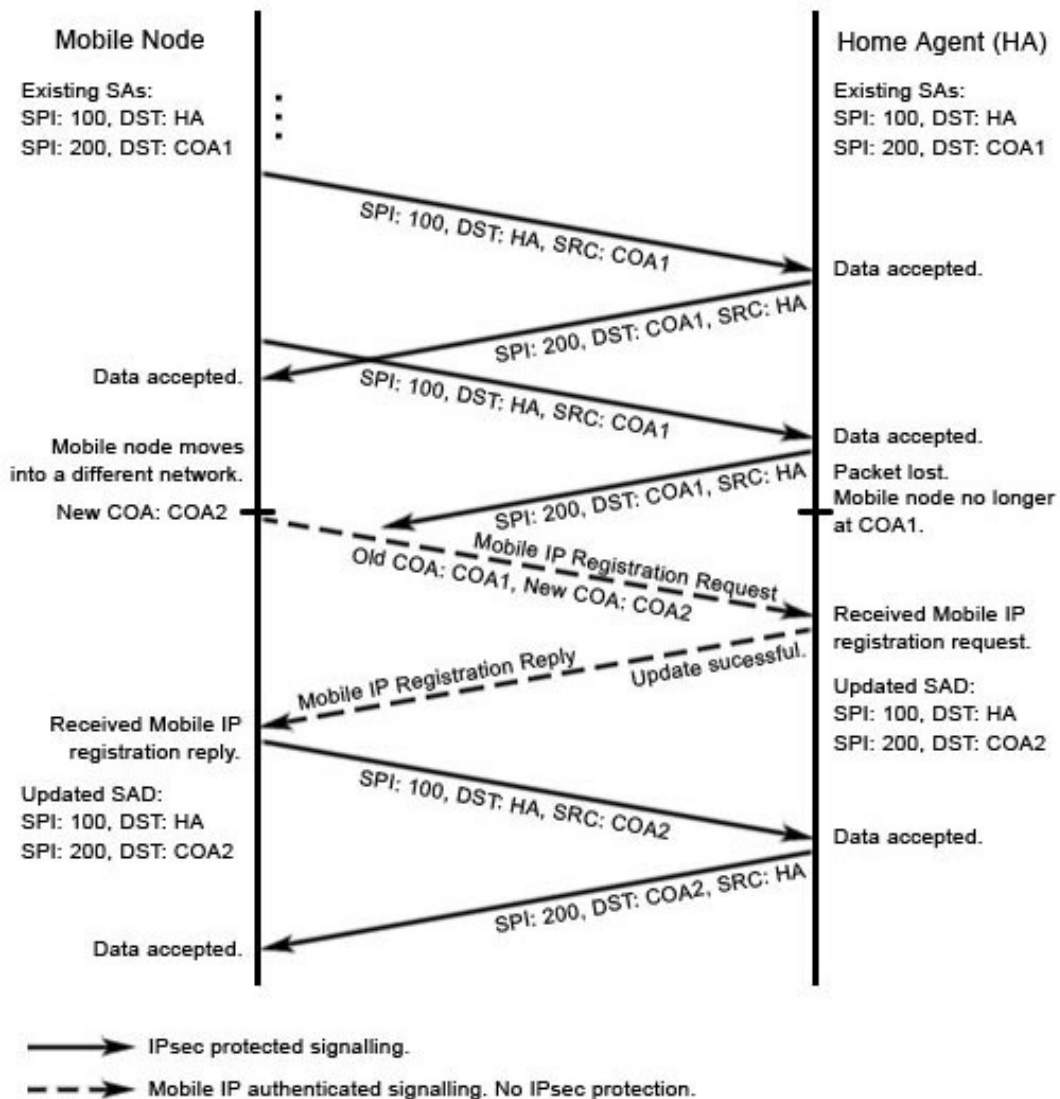


Figure 4.4: The IPsec tunnel handoff process when using Mobile IP registration messages for dynamic updates of the IPsec tunnel endpoints.

# Chapter 5

## Testbed Implementation

In order to test the feasibility of our proposed solution and to study the performance of the various approaches, we designed and developed a testbed for evaluating the various solutions under different handoff conditions. An isolated network is used in order to provide a controlled environment under which various performance factors, such as the bandwidth overhead, as well as the setup and handoff latencies can be studied.

### 5.1 Network Configuration

For the purpose of our experiments, the testbed is developed on an isolated Institute of Electrical and Electronics Engineers (IEEE) 802.11b wireless local area network. The testbed consists of three machines and three wireless access points to enable the testing of the IPsec tunnel handoffs. Figure 5.1 shows how the network is set up. The device specifications and configurations are listed in Table 5.1. Both the home agent and the mobile node are configured to run Dynamics Mobile IP and Racoon *Internet Key Exchange*<sup>1</sup> (IKE) key management daemon. The configuration files for these services are included in Appendices B and C, respectively. A router is set up in the foreign network to create two foreign networks: IPSEC-FN1 and IPSEC-FN2. The foreign router is necessary to ensure proper operation of Mobile IP. Due to limited resources, IPSEC-HN and IPSEC-FN1 are actually the same device with different configurations loaded. This does not have significant impact on the experiments, as only two wireless access points are required at any one time for all experiments in this research. The wired Ethernet network operates at 100 Mbits/sec while the wireless IEEE 802.11b network operates at 11 Mbits/sec. All security services on the wireless network are disabled. The network is lightly loaded in order to study the optimal performance of the various approaches. This also reduces the fluctuations in the performance measurements. While this may not be indicative of the performance in an actual

---

<sup>1</sup>The Internet Key Exchange (IKE) is an IPsec standard protocol used to ensure security for VPN negotiation and remote host or network access. Specified in IETF RFC 2409 [7], IKE defines an automatic means of negotiation and authentication for IPsec SAs. The IKE protocol ensures security for SA communication without the pre-configuration that would otherwise be required.

network, it still allows us to compare the performance of the various approaches.

Three setup scripts, `ha-setup`, `fa-setup`, and `mn-setup` were written to configure the basic networking functionalities on the home agent, the foreign router, and the mobile node, respectively. The Dynamics Mobile IP and Racoon IKE key management daemon are disabled in the basic network configuration. These services may be started, when necessary, by other means. The setup scripts also restore the systems to a consistent starting point for all the experiments. All three scripts are included in Appendix D.

## 5.2 Solution Implementation

In order to compare the performance of the various approaches described in Section 4, we implemented three different solutions for securing network traffic in a mobile environment. In this report, the solutions will be referred to as follows:

- **No IPsec** refers to the basic network configuration with no security services applied. This provides a baseline for studying the performance of the different solutions.
- **IPsec** refers to a standard IPsec setup. This solution is based on the *road warrior*<sup>2</sup> configuration suggested in the IPsec HOWTO [18]. The road warrior configuration is currently used for mobile nodes who are using unknown dynamic IP addresses to connect to a VPN gateway. With this approach, new IPsec tunnels have to be negotiated when the mobile node moves into a different network.
- **MIPsec** refers to our implementation of IPsec over Mobile IP. The PSU Mobile IP implementation is unsuitable for this experiment as its home agent daemon only runs on FreeBSD systems. The home agent daemon is a key component of Mobile IP, and the mobile node cannot function without it. The solution to this problem is likely to require that part, if not the whole, of our testbed be converted to run FreeBSD. Due to lack of experience with FreeBSD systems, we determine that it was simpler to develop our own Linux implementation, which would be sufficient for the purpose of our experiments.
- **DynIPsec** refers to the implementation of our proposed solution of using Mobile IP registration messages for dynamic updates of the IPsec tunnel endpoints.

In an effort to simplify and minimise the development time, we implemented the solutions as shell scripts and simple programs, which make use of open source IPsec and Mobile IP distributions. More specifically, the implementations are based on Racoon [3] and Dynamics Mobile IP [1]. Racoon is an IKE key management daemon used to provide automated IPsec key negotiation functionality and is distributed as part of the IPsec-tools package [2]. The IPsec-tools package is an open source IPsec implementation. The package provides the default IPsec driver for the Fedora distribution. In addition to the IPsec

---

<sup>2</sup>Road warriors are clients using unknown dynamic IP addresses to connect to a VPN gateway.

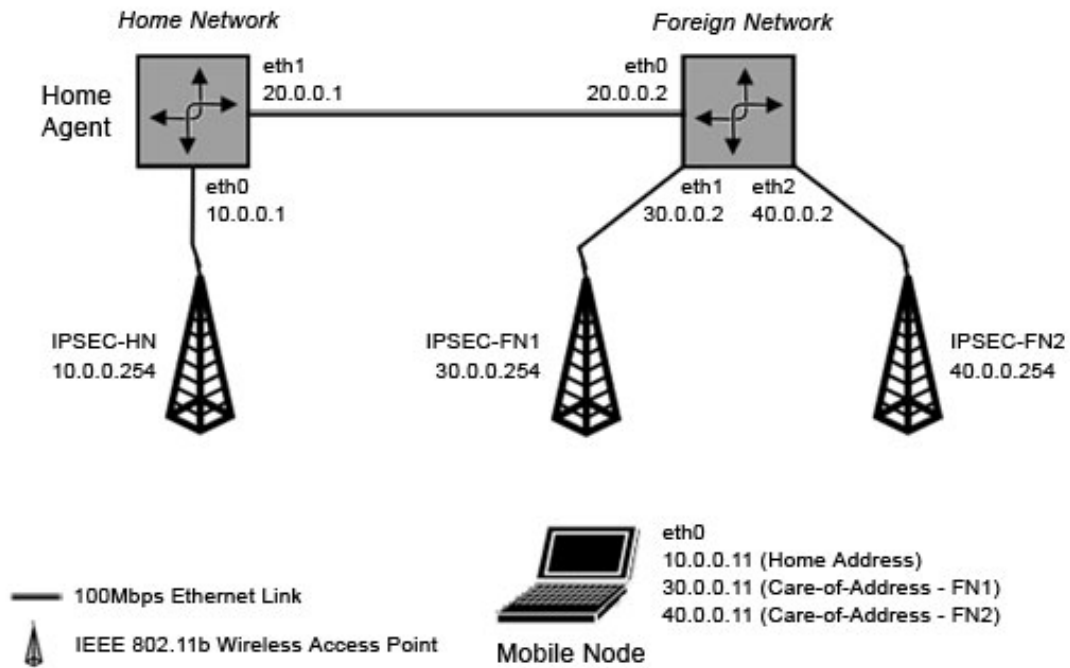


Figure 5.1: Testbed network configuration.

Device	Specifications and Configurations
Home Agent	Intel Pentium III 500MHz 512MB RAM Fedora Core 3, Kernel Version 2.6.9-1.667
Mobile Node	Intel Pentium III 500MHz 512MB RAM Fedora Core 3, Kernel Version 2.6.9-1.667 Lucent WaveLAN Turbo Silver 11Mb Card
Foreign Router	AMD Athlon 800MHz 768MB RAM Fedora Core 3, Kernel Version 2.6.9-1.667
IPSEC-HN/ IPSEC-FN1	Lucent WavePOINT-II Enterasys Card Noise: -96; Signal: -46; Quality: 50/92 * The configuration files are included in Appendix A.
IPSEC-FN2	Lucent Orinoco AP-2000 V2.4.4 Lucent Orinoco Gold Card Noise: -90; Signal: -46; Quality: 46/92

Table 5.1: Device specifications and configurations.

driver and the Racoon IKE daemon, the package also provides a tool called **setkey**, which enables users to manually add, update, flush, and dump the contents of the SPD and SAD. On the other hand, Mobile IP functionality is provided by Dynamics Mobile IP. Dynamics Mobile IP is an open source project. Even though there has been little recent developments, Dynamics Mobile IP is used in various mobile networking test environments and is proven to be stable. Additionally, the Dynamics implementation runs entirely on user space and does not require any modifications to the kernel. Dynamics Mobile IP provides both the home agent, **dynhad**, and the mobile node daemon, **dynmnd**, required for this study. The package also includes **dynha\_tool** and **dynmn\_tool**, which provide an interface to the home agent and the mobile node daemon, respectively. Both of these packages are widely available and are proven to be stable. While using existing software packages simplifies the development, it also restricts the implementation.

On the home agent, the implementations are separated into three scripts: **ha-ipsec**, **ha-mipsec**, and **ha-dynipsec**. As the names suggest, the scripts are for standard IPsec, MIPsec, and DynIPsec, respectively. These scripts are automated and are capable of handling a single mobile node. The scripts on the mobile node are similarly organised but are further separated according to the handoff condition. The three different handoff scenarios are:

- **setup** - when the mobile node moves from the home network into a foreign network, or when the mobile node first requires IPsec services in a foreign network;
- **handoff** - when the mobile node moves between foreign networks; and
- **teardown** - when the mobile node moves from a foreign network into the home network, or when IPsec services are no longer required.

In practice, another script is required on the mobile node to determine the most appropriate time for a handoff, and to execute the appropriate handoff script accordingly. All the home agent and mobile node scripts are included in Appendices E, F, and G.

### 5.2.1 Disabled Path MTU Discovery

Path MTU discovery [14] is a protocol used to dynamically discover the largest datagram size that does not require fragmentation anywhere along the path from the source to the destination. Path MTU discovery does not function correctly when IPsec tunnelling and Mobile IP tunnelling are used concurrently. When the IPsec tunnel is used together with the IP in IP (IP-IP) tunnel set up by Mobile IP, path MTU discovery results in a recursive loop, which continually reduces the MTU value. This seems to be a limitation of the IPsec protocol when attempting to identify the originating host for use in further propagation of the path MTU information, and is further addressed in RFC 2401 [12]. Therefore, we decided to disable path MTU discovery on both the home agent and the mobile node.

```
echo 1 > /proc/sys/net/ipv4/ip_no_pmtu_disc
```

### 5.2.2 Forced Handoff

For practical reasons, instead of physically moving the mobile node to trigger a handoff, a script is used to simulate the handoff by forcing the mobile node to associate with the target access point, as shown below.

```
ifconfig $DEF_IF down
iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $MN_COA_IP up

route add default gateway $GATEWAY
```

First, the interface is taken down in order to remove all the routing entries associated with the interface. Then, the interface is configured using `iwconfig` with the *ESSID*<sup>3</sup> of the target network before it is brought up with the new co-located CoA. Once completed, the default route is manually added using the `route` command. Currently, the user is required to provide the ESSID, the new co-located CoA, and the default gateway in order to complete the handoff. In practice, these values can be pre-configured, or determined automatically during handoff using various means, such as Dynamic Host Configuration Protocol (DHCP) requests and Internet Control Message Protocol (ICMP) router solicitation messages.

### 5.2.3 Use of Home Address as the Mobile Node's Source IP Address

According to RFC 3344 [16], the mobile node uses its home address as the source address of all IP packets that it sends, except for datagrams sent for certain mobility management functions. However, when operating in *full direct HA tunnel mode*<sup>4</sup>, Dynamics Mobile IP uses the mobile node's CoA for all IP packets created on the mobile node. In order to apply the appropriate IPsec functions, the mobile node is required to use its home address. Therefore, we configured the mobile node to use advanced routing in order to force the mobile node to use its home address instead of the CoA. With MIPsec, a new IP-IP tunnel is created on the mobile node.

```
ip tunnel add $AR_TUN local $MN_COA_IP remote $HA_EXT_IP dev $DEF_IF mode ipip
ifconfig $AR_TUN $MN_HOM_IP up
ip rule add to $HA_EXT_IP table $AR_ID
ip route add default dev $AR_TUN table $AR_ID
```

The new tunnel is similar to the original Mobile IP tunnel, using the co-located CoA as the local address. Once set up, the tunnel is brought up using the home address. This technique forces the mobile node to use the home address as the source IP address when creating new IP packets. Advanced routing is then used to configure the new tunnel as the

---

<sup>3</sup>The Extended Service Set ID (ESSID) is the identifying name of a wireless network. It allows one network to be clearly distinguishable from another.

<sup>4</sup>The full direct HA tunnel mode refers to Mobile IP's co-located CoA mode with reverse tunnelling.

default route for packets destined for the home agent. DynIPsec uses a similar solution to force the mobile node to use its home address. In this case, no new tunnel is created as DynIPsec does not require the additional IP-IP tunnelling. Instead, a high priority default route is used.

```
ip rule add from 0.0.0.0 table $AR_ID
ip route add default dev $DEF_IF src $MN_HOM_IP via $GATEWAY table $AR_ID

iptables -t nat -A POSTROUTING -s $MN_HOM_IP -j SNAT --to-source $MN_COA_IP
```

The new default route forces the mobile node to create new IP packets using its home address. Using `ip rule`, the new route is assigned a higher priority so that it takes precedence over the standard default route, which is required for general routing purposes. A problem arises when no additional IPsec services is applied to the IP packets, as is the case with Mobile IP signalling. Without the additional IPsec tunnel, the IP packets will have the home address in the outermost IP header. Depending on the network configuration, these packets may be dropped. To resolve this problem, network address translation is used to ensure that the outermost IP header uses the co-located CoA.

#### 5.2.4 Split Tunnelling for the Address Change Notification System

When the mobile node's CoA changes, the SAD and SPD entries on the home agent and the mobile node become invalid. As a result, Mobile IP registration packets cannot be sent through the IPsec tunnel during handoff, as the registration reply from the home agent cannot be decrypted by the mobile node. To overcome this problem, we employed split tunnelling for the Mobile IP signalling. Split tunnelling is a technique which allows certain pre-defined traffic, to and from the host, to by-pass the IPsec services.

```
spdadd 0.0.0.0/0[any] $HA_EXT_IP[434] any -P out none;
spdadd $HA_EXT_IP[434] 0.0.0.0/0[any] any -P in none;
```

These two additional SPD entries exclude the Mobile IP signalling from getting encrypted. The alternative solution is to remove the dependence on IP destination address for uniquely identifying an SA, as previously described in Section 4.2.1.

#### 5.2.5 Firewall Rules for Blocking Insecure Communication during Handoff

Due to the IPsec security policy configuration with standard IPsec and MIPsec, IP packets may “leak” unencrypted during handoff. To overcome this problem, the firewall is configured to block all traffic leaving the mobile node, except Mobile IP signalling, until the necessary updates are completed.

```
iptables -A OUTPUT -s $MN_OLD_IP -p udp --destination-port ! 434 -j DROP
iptables -A OUTPUT -p icmp -j DROP
```

Currently, with all three approaches, traffic to and from the mobile node during the setup process may not be secure until the setup script completes execution. The technique of using firewall rules for blocking insecure traffic can also be applied in this case to prevent the problem from occurring.

### 5.2.6 Updating the SAD and SPD

With DynIPsec, both the home agent and the mobile node have to update their SAD and SPD during handoffs in order to maintain the existing SAs. The SAD and SPD are usually located in a secure location, such as the kernel, and is not directly accessible to users for obvious security reasons. The `setkey` tool, provided as part of the IPsec-tools package [2], enables users to manipulate the SAD and SPD. However, it does not provide the capability to modify existing SA entries. Therefore, we developed a simple program, `upaddr`, which interfaces through `setkey` to perform the necessary changes to the SAD and SPD. The source code for `upaddr` is included in Appendix H. The program removes the existing IPsec tunnel and creates a new identical tunnel using the new IP address, effectively “updating” the SAs. Due to limitations of `setkey`, it was not possible to retain the life time duration parameters of the existing tunnel. This effectively resets the life time duration value every time the IPsec tunnel is handed off, potentially delaying the IPsec tunnel re-keying process. All the other tunnel parameters, including the IPsec encryption keys, are retained.

As the tunnel is being “updated”, `racoona` may try to negotiate a new IPsec tunnel for the old co-located CoA. This is a limitation with the Racoon design. The Racoon implementation, as with most IPsec implementations, are designed for a fixed network. IP address updates on the SAD and SPD does not usually occur in a fixed network environment. To prevent this situation from occurring, it is necessary to *kill* the `racoona` service prior to performing the address update.

```
kill -9 $(ps -e | grep racoon | awk '{print $1}') > /dev/null 2>&1
rm -f /tmp/.racoon
```

Simply stopping the service does not work, as it removes the SAD entries, thus effectively breaking the IPsec tunnel.



# Chapter 6

## Performance Study

The aim of this section is to study and compare the performance of the various approaches. The performance of our proposed solution is compared against the current approach based on re-negotiating new IPsec tunnels, and the alternative approach of running IPsec over Mobile IP. The various approaches are compared in terms of bandwidth overhead and handoff latency.

### 6.1 Bandwidth Overhead

Bandwidth overhead is a key measure in determining the performance of a given solution, as it is inversely related to the actual data transfer rate. Both IPsec and Mobile IP introduce additional protocol headers, which increase the bandwidth overhead. Table 6.1 shows the minimal header size of the various protocols used in this study. In practice, the actual header size may be higher due to various factors, including the use of options, padding, and encryption.

Protocol	Minimal Header Size (bytes)
Ethernet	14
IP	20
ESP	30
TCP	20

Table 6.1: The minimal header size of several common protocols.

#### 6.1.1 Ping Latency

Before studying the bandwidth overhead, it may be worthwhile to briefly consider the processing overhead of the various approaches. Studying the *ping*<sup>1</sup> latencies will provide

---

<sup>1</sup>Ping is a basic Internet program that lets you verify that a particular IP address exists and can accept requests. Ping is used diagnostically to ensure that a host computer you are trying to reach is actually

some insights on the processing overhead and the routing delays introduced by the different solutions. We measured the ping latency by taking the average of 100 pings sent at one second intervals.

```
MN > ping -c 100 10.0.0.1
```

The result of this experiment is summarised in Table 6.2. After adding IPsec, the ping round-trip time (RTT) is approximately 0.6 milliseconds higher. The result is fairly similar in the case of DynIPsec due to the IPsec processing required at both ends of the IPsec tunnel. In addition to the IPsec processing, MIPsec also requires the packets to be encapsulated and decapsulated due to the Mobile IP tunnelling, which results in the highest ping latency of all approaches, with an average RTT of 3.2 milliseconds.

Solutions	Min (ms)	Avg (ms)	Max (ms)
No IPsec	2.279	2.379	2.813
IPsec	2.949	3.056	3.655
MIPsec	3.092	3.226	3.756
DynIPsec	2.963	3.088	3.629

Table 6.2: The ICMP ping round trip time between the mobile node and the home agent.

### 6.1.2 MN-HA Bandwidth Overhead

In order to study the actual bandwidth overhead, we use a bandwidth measurement tool called `iperf` [15]. `iperf` uses a client-server architecture. It works by having the client send a continuous stream of random data to the server in an attempt to use up all available bandwidth for a given length of time. Based on the amount of data sent, the client is then able to calculate the available bandwidth.

The goal for the first part of this experiment is to measure the bandwidth from the mobile node to the home agent (MN-HA). In this case, the `iperf` server is set up on the home agent. The mobile node then tries to measure the bandwidth while it is connected through a foreign network, specifically IPSEC-FN1.

```
MN > iperf -t 60 -m -c 10.0.0.1
```

Table 6.3 compares the *maximum segment size*<sup>2</sup> (MSS) of the various approaches, as reported by `iperf`, when using the default MTU of 1500 bytes. The results show that applying IPsec reduces the MSS by 56 bytes. DynIPsec shows a similar reduction as it uses the same protocol headers as standard IPsec. MIPsec shows the highest overhead of

---

operating. Ping can also be used with a host that is operating to see how long it takes to get a response back.

<sup>2</sup>The maximum segment size (MSS) is the largest amount of data, specified in bytes, that a computer or communications device can handle in a single, unfragmented piece.

all approaches, reducing the MSS to 1376 bytes. This is due to the additional 20 bytes IP header introduced by the additional Mobile IP tunnelling. In this case, the MSS value when running MIPsec is actually only 16 bytes lower when compared to that of standard IPsec due to data alignment restrictions introduced by the encryption algorithm.

Solutions	MSS (bytes)
No IPsec	1448
IPsec	1392
MIPsec	1376
DynIPsec	1392

Table 6.3: The Maximum Segment Size (MSS) of the various approaches with the default MTU of 1500 bytes.

The `iperf` measurements for the MN-HA bandwidth is shown in Table 6.4. The overhead for a given approach is derived simply by calculating the reduction in measured bandwidth with respect to the no IPsec approach. A few of the overhead calculations may seem slightly off due to rounding of the bandwidth measurements. The results show that the maximum bandwidth when using a basic network configuration is approximately 5.03 Mbits/sec. With standard IPsec and DynIPsec, the reduction in available bandwidth is approximately 0.22 Mbits/sec, or 4.4 percent. As expected, MIPsec shows the highest overhead of all, reducing the available bandwidth by approximately 6.1 percent.

The size of the protocol headers remains the same regardless of the packet sizes. Consequently, the overhead increases as the packet size decreases. In order to study the effect of smaller packet sizes on the bandwidth overhead, the MTU on the home agent and the mobile node is lowered to 576 bytes. The MTU of an interface can be set using `ifconfig` as follow:

```
MN > ifconfig eth0 mtu 576
```

We chose an MTU of 576 bytes as it is a value commonly used by older devices. When multiple devices along a path have varying MTU values, the lowest value should be used to prevent fragmentation. Fragmentation should be avoided where possible, as it adds additional overhead which further reduces the available bandwidth. The MTU value is usually determined automatically using path MTU discovery [14]. The result of the experiment is summarised in Table 6.5. It shows that the bandwidth overhead with an MTU setting of 576 bytes is significantly higher when compared to that at 1500 bytes. In this case, the percentage overhead is approximately 12 percent with IPsec and DynIPsec, and 17 percent with MIPsec.

### 6.1.3 HA-MN Bandwidth Overhead

Having studied the MN-HA bandwidth overhead, the experiment is repeated in the reverse direction with the `iperf` server now running on the mobile node. This allows the bandwidth

from the home agent to the mobile node (HA-MN) to be measured. As the home agent has two network interfaces, `iperf` has to be explicitly bound to the home agent's internal interface using the `-B` option as follow.

```
HA > iperf -t 60 -m -B 10.0.0.1 -c 30.0.0.11
```

Table 6.6 and Table 6.7 show the HA-MN bandwidth measurements when using an MTU setting of 1500 bytes and 576 bytes, respectively. Theoretically, these measurements should be similar. However, the HA-MN bandwidth measurements are consistently between 0.1 and 0.4 Mbits/sec higher than the corresponding MN-HA measurement. Since the same situation occur when running the basic network configuration, the discrepancy is unlikely to be caused by the IPsec and Mobile IP configurations. This suggests the most likely cause to be the difference in the hardware capability of the wireless access point and the wireless network interface on the mobile node. Due to limited hardware resources, we were unable to verify the cause of this discrepancy.

Although the HA-MN bandwidth measurements are slightly higher, the percentage overhead is still similar when using an MTU setting of 576 bytes. However, at the default MTU setting of 1500 bytes, the results become less consistent. Upon further analysis, it was found that the Transmission Control Protocol (TCP) acknowledgements (ACKs) from the mobile node seems to be delayed, thus preventing the home agent from sending any further data, which consequently leads to a reduced transfer rate. We also found that the problem only occurs when the network is heavily loaded. The `iperf` bandwidth measurement process saturates the network with IP packets, effectively loading the network to its maximum capacity. When tested using small File Transfer Protocol (FTP) file transfers, the problem did not occur. This suggests that the problem lies in the queueing and buffering mechanism of the wireless devices when operating under heavy load conditions.

#### 6.1.4 FTP Bandwidth and Transfer Time

An actual FTP data transfer was also carried out to test the validity of the `iperf` measurements. The test was conducted using the `vsftpd` server and the `gFTP` client. Both of these applications are included in the standard Fedora Core 3 distribution. Due to the inconsistent HA-MN bandwidth measurements obtained previously, we decided to set up the FTP server on the mobile node. For this experiment, a 256 MB binary file (ISO image) is transferred from the mobile node to the home agent using the default MTU of 1500 bytes. The bandwidth measurements, as reported by `gFTP`, and the transfer time for the various approaches are shown in Table 6.8. The transfer time can be calculated from the bandwidth measurements and was checked by actually timing the transfers.

The `gFTP` bandwidth measurements are similar to those obtained using `iperf`. Additionally, the results also show that MIPsec takes approximately 8 seconds longer to transfer 256 MB of data when compared to standard IPsec and DynIPsec.

<b>Solutions</b>	<b>Bandwidth (Mbits/sec)</b>	<b>Overhead (Mbits/sec)</b>	<b>Percentage Overhead</b>
No IPsec	$5.03 \pm 0.01$	-	-
IPsec	$4.81 \pm 0.02$	$0.22 \pm 0.03$	$4.4 \pm 0.5$
MIPsec	$4.72 \pm 0.03$	$0.31 \pm 0.04$	$6.1 \pm 0.7$
DynIPsec	$4.81 \pm 0.01$	$0.22 \pm 0.02$	$4.4 \pm 0.5$

Table 6.4: The MN-HA bandwidth measurements for the various approaches with the default MTU value of 1500 bytes.

<b>Solutions</b>	<b>Bandwidth (Mbits/sec)</b>	<b>Overhead (Mbits/sec)</b>	<b>Percentage Overhead</b>
No IPsec	2.66	-	-
IPsec	2.34	0.32	12
MIPsec	2.22	0.44	17
DynIPsec	2.34	0.32	12

\* The bandwidth measurement error is less than 0.01 Mbits/sec.

Table 6.5: The MN-HA bandwidth measurements for the various approaches with a lowered MTU value of 576 bytes.

<b>Solutions</b>	<b>Bandwidth (Mbits/sec)</b>	<b>Overhead (Mbits/sec)</b>	<b>Percentage Overhead</b>
No IPsec	$5.24 \pm 0.03$	-	-
IPsec	$5.17 \pm 0.02$	$0.07 \pm 0.04$	$1.3 \pm 0.8$
MIPsec	$5.02 \pm 0.01$	$0.22 \pm 0.04$	$4.1 \pm 0.7$
DynIPsec	$5.18 \pm 0.03$	$0.05 \pm 0.06$	$1.0 \pm 1.2$

Table 6.6: The HA-MN bandwidth measurements for the various approaches with the default MTU value of 1500 bytes.

<b>Solutions</b>	<b>Bandwidth (Mbits/sec)</b>	<b>Overhead (Mbits/sec)</b>	<b>Percentage Overhead</b>
No IPsec	$2.83 \pm 0.01$	-	-
IPsec	$2.50 \pm 0.01$	$0.33 \pm 0.01$	$12 \pm 1$
MIPsec	$2.36 \pm 0.01$	$0.48 \pm 0.01$	$17 \pm 1$
DynIPsec	$2.51 \pm 0.01$	$0.32 \pm 0.02$	$11 \pm 1$

Table 6.7: The HA-MN bandwidth measurements for the various approaches with a lowered MTU value of 576 bytes.

<b>Solutions</b>	<b>Bandwidth (Kbytes/sec)</b>	<b>Transfer Time (sec)</b>
No IPsec	$613.3 \pm 0.2$	$427 \pm 0.1$
IPsec	$585.9 \pm 0.5$	$447 \pm 0.3$
MIPsec	$576.3 \pm 0.9$	$455 \pm 0.7$
DynIPsec	$586.2 \pm 1.3$	$447 \pm 1.0$

Table 6.8: The FTP bandwidth and transfer time for the various approaches.

## 6.2 Handoff Latency

The handoff latency is the length of time during handoff when the home agent and the mobile node is unable to communicate securely. High handoff latencies causes noticeable delays for the end users, and may even result in connection time-outs. The factors affecting the handoff latency include:

- the number of message exchanges required;
- the amount of processing involved on the end nodes;
- the speed of the networks; and
- the network load.

Consequently, the results are dependent on the machine capabilities and the network configuration.

The number of messages exchanged during the tunnel handoff is a main factor affecting the handoff latency. These messages have to travel through various networks, including the foreign wireless network and the Internet, adding to the handoff delay. Typically, between three to six round trips are required to setup an IPsec tunnel. This is similar for all approaches, although an additional round trip is required with IPsec over Mobile IP and our proposed solution for the Mobile IP registration messages. However, during handoff, the standard IPsec approach based on re-establishing new IPsec tunnels still requires between three to six round trips. In contrast, IPsec over Mobile IP and our proposed solution only require a single round trip. On top of that, the processing delays on the end nodes also have to be taken into account. Delays occur in various forms, such as the time required to update the SAD and SPD, as well as the delay in updating the routing caches.

The following experiment aims to compare the actual handoff latencies of the various approaches under different handoff conditions. Specifically, the setup, handoff, and teardown latencies are measured under the following conditions:

- setup - the mobile node moves from the home network, IPSEC-HN, to a foreign network, IPSEC-FN2;
- handoff - the mobile node moves from a foreign network, IPSEC-FN1, to another foreign network, IPSEC-FN2; and

- teardown - the mobile node moves from a foreign network, IPSEC-FN2, to the home network, IPSEC-HN.

All the measurements in this experiment, unless otherwise stated, are taken between the mobile node and the home agent, with the home agent serving the role of the correspondent node. In a typical situation where the correspondent node is a machine other than the home agent, the handoff latencies will be higher due to the additional time required for the IP packets to be routed between the home agent and the correspondent node. However, if present, this additional delay will be the same in all cases.

The script execution time is not an appropriate measure of the handoff latency as it does not include various delays, such as the time required to negotiate an IPsec tunnel, the delay in updating the routing cache, and the time for the home agent to perform the necessary updates. Without analysing the actual traffic to and from a given host, it is difficult to determine when the handoff is actually completed. Therefore, we decided to measure the handoff latency using a continuous stream of ICMP ping requests sent at regular intervals. The handoff latency is measured by capturing and analysing the network traffic at the mobile node using a network protocol analyser, *ethereal*. For the purpose of this experiment, the handoff latency is defined as the length of time between the receipt of the first “well-formed” ping reply in the destination network and the last ping reply in the origin network. In this case, a “well-formed” packet is one which has the proper IPsec services and where appropriate, the necessary Mobile IP encapsulation applied to it. The stream of ping packets will undoubtedly increase the network load and consequently, the handoff latency. The network protocol analyser on the mobile node also introduces an additional delay due to the extra processing load on the system. However, these additional delays will be consistent for all the latency measurements. In order to provide an accurate measurement, the ping interval is set to 0.1 second as follows.

```
MN > ping -i 0.1 -s 8 10.0.0.1
```

Additionally, the ping data size is reduced to 8 bytes to minimise the extra network load. The results of the experiment are summarised in Table 6.9. The large error on the MIPsec setup time is caused by occasional route selection delays on the home agent. Since the IPsec tunnel is setup using the mobile node’s home address, the home agent assumes that the mobile node is still at home even though the mobile node has actually moved into a foreign network. This results in an approximately 10 second delay as the home agent attempts to locate the mobile node in the home network. In this situation, there may be a kernel parameter which determines how long a given host should wait for a reply. However, we are unable to determine the exact parameter at this point.

Out of the three solutions, MIPsec has the lowest handoff latency of approximately 1.3 seconds. However, it also has the highest setup and teardown latency. Our proposed solution has a lower handoff latency compared to that of standard IPsec. Interestingly, standard IPsec’s handoff latency of 12.2 seconds is actually higher than its own setup time. Theoretically, these two measurements should be similar since both involves the negotiation of a new IPsec tunnel. Further investigation reveals that the high latency is

Solutions	Handoff Latency (sec)		
	Setup	Handoff	Teardown
IPsec	$2.9 \pm 0.1$	$12.2 \pm 0.1$	$1.7 \pm 0.2$
MIPsec	$5.9 \pm 8.5^\dagger$	$1.3 \pm 0.1$	$2.6 \pm 0.4$
DynIPsec	$4.5 \pm 1.0$	$2.3 \pm 0.5$	$2.3 \pm 0.1$

<sup>†</sup>Route selection delay on the home agent.

Table 6.9: The handoff latency for the various approaches.

caused by an IPsec tunnel negotiation time-out. This is most likely a limitation of the Racoon implementation when dealing with IP address changes. Arguably, Racoon was not designed to work in a mobile environment. However, DynIPsec's handoff latency of approximately 2.3 seconds is still lower on average, when compared to the setup time of standard IPsec.

### 6.2.1 Pre-shared Key vs. RSA Signature Authentication

All the handoff latencies in the previous experiment are based on pre-shared key phase 1 authentication. The aim of this next experiment is to compare the difference in the IPsec tunnel setup time when using pre-shared key authentication versus RSA signature authentication. In this case, to use RSA signature authentication, both the home agent and the mobile node are configured to use *X.509*<sup>3</sup> certificates. X.509 certificates can be generated using the `openssl` tool available as part of the standard Fedora Core 3 distribution. In addition, the Racoon configuration file on the home agent has to be modified to include a `certificate_type` directive, as well as to update the `authentication_method` as shown below.

```
remote anonymous {
    exchange_mode main;
    passive on;
    certificate_type x509 "hacert.pem" "hakey.pem";
    verify_cert off;
    generate_policy on;
    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method rsasig;
        dh_group modp1024;
    }
}
```

Similar changes are also required on the mobile node.

---

<sup>3</sup>X.509 is a widely used standard for defining digital certificates. X.509 is actually an ITU Recommendation, which means that it has not yet been officially defined or approved for standardised usage.



```

remote 20.0.0.1 {
    exchange_mode main;
    verify_cert off;
    certificate_type x509 "mncert.pem" "mnkey.pem";
    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method rsasig;
        dh_group modp1024;
    }
}

```

Table 6.10 shows the setup time difference between using pre-shared key authentication and RSA signature authentication as a mobile node tries to establish an IPsec tunnel from a foreign network, IPSEC-FN1. These setup times are lower compared to previous measurements as no handoff is involved in this case. The results show that using RSA signature authentication for setting up IPsec tunnels takes on average 0.4 seconds longer compared to pre-shared key authentication. This difference in setup time applies to all the setup latency measurements in the previous experiment. Additionally, it also applies to the handoff latency of standard IPsec as it involves the negotiation of a new IPsec tunnel.

Authentication Method	Setup Time (sec)
Pre-shared Key	$2.1 \pm 0.8$
RSA Signature	$2.5 \pm 0.6$

Table 6.10: The IPsec tunnel setup time.

### 6.3 Maintaining FTP File Transfer During Handoff

An FTP file transfer was conducted to test whether the various approaches are able to maintain connection-oriented services running over IPsec during the tunnel handoffs. The test is conducted by first setting up a large FTP file transfer in the origin network. Then, as the FTP file transfer is running, the mobile node is moved into another network. The experiment is repeated for all the different handoff scenarios. The FTP file transfer is maintained in all cases with both MIPsec and DynIPsec but consistently stalled with the standard IPsec setup. With the standard IPsec setup, the transfer stalled for approximately 180 seconds before the transfer resumed. Upon further investigation, it was found that two new TCP connections for the FTP data and control lines had been established to resume the transfer, while the old connections were left in a `FIN_WAIT1` state. The `FIN_WAIT1` state indicates that the socket is closed, and the connection is shutting down. There are several kernel parameters which influences the TCP connection timeout delay. In particular, the `tcp_keepalive_time` specifies how often TCP sends out keep-alive messages, and the

`tcp_keepalive_probes` determines the number of keep-alive probes that TCP should send before it decides that the connection is broken [6]. These parameters can be configured by the root user but will affect all TCP connections on the system. The default TCP keep-alive time on the system is two hours. This suggests that the timeout, in this case, is built into the `gFTP` client. Regardless of the TCP connection timeout situation, standard IPsec is unable to maintain TCP connections across handoff due to the change in IP address.

# Chapter 7

## Security Implications

It is important that adding mobility support to IPsec does not introduce any new security vulnerabilities. Our proposed solution does not require any modifications to the IPsec standard. Therefore, it should be as secure as a standard IPsec approach. However, the use of Mobile IP registration messages for updating the IP address parameters in the SAD and SPD does require the security of the Mobile IP registration process to be considered. RFC 3344 [16] states that all Mobile IP registration messages between a mobile node and its home agent must be authenticated with an authorisation-enabling extension. This is necessary to eliminate problems which result from the uncontrolled propagation of remote redirects. In Dynamics Mobile IP implementation, the registration messages are authenticated using either a pre-shared key, or an Authentication, Authorisation and Accounting (AAA) infrastructure.

Should an attacker somehow manage to compromise the Mobile IP authentication mechanism, the IPsec tunnel can potentially be redirected using a spoofed registration message. However, the encrypted data will still remain secure since the attacker does not have the IPsec key necessary to decrypt the data. From the point of view of the user, this would be a denial-of-service attack. Running IPsec over Mobile IP suffers from a similar vulnerability due to the use of Mobile IP tunnels.

# Chapter 8

## Areas for Future Research

This section identifies several potential topics for future researches.

- One potential research topic is to study the effect of compression and the different tunnelling protocols on the bandwidth overhead of IPsec over Mobile IP. In the handoff latency experiment, our implementation of IPsec over Mobile IP, MIPsec, has the lowest handoff latency when compared against standard IPsec and DynIPsec. However, it also has the highest bandwidth overhead out of all the different approaches. There are two general techniques that may be employed to minimise this problem. The first technique involves the use of compression on the protocol headers and the data payload using algorithms such as the Van Jacobsen TCP/IP header compression and IPcomp for compressing the IPsec data payload. The second technique seeks to minimise the size of the additional IP header introduced by the IPsec tunnelling and the Mobile IP tunnelling. This involves using alternative tunnelling protocols, such as minimal encapsulation or GRE tunnelling, which uses a smaller header compared to the standard IP-IP tunnelling.
- Another interesting topic would be to extend the work done in this research to a 3G wireless WAN. Currently, the solutions have been implemented and tested on an IEEE 802.11 wireless LAN. IEEE 802.11 wireless LANs and 3G wireless WANs have complementary features and combine to offer high throughput and ubiquitous coverage. IEEE 802.11 wireless LANs provide mobile users within the coverage areas with high throughput, while 3G wireless WANs enable the users to stay connected as they move out of these localised coverage areas. Extending the solution to interoperate between IEEE 802.11 wireless LANs and 3G wireless WANs would enable mobile users to seamlessly handoff IPsec tunnels between these two network technologies.
- The focus of this research is on developing mobility support for IPsec in IPv4 networks. It would be interesting to study how this research might extend to IPv6 networks. The IPv6 standard, as defined in RFC 2460 [5], has several enhancements over IPv4. The two features of interest are IPv6's built-in mobility support, and the IPv6 IPsec AH and ESP header extensions.

# Chapter 9

## Conclusions

There are two general approaches to achieve seamless IPsec tunnel handoffs when moving between different networks: to run IPsec over Mobile IP, or to dynamically update the IPsec tunnel endpoints. As part of this research, we proposed a variation of the latter approach whereby Mobile IP registration messages are used to trigger an update of the SAD and the SPD. A prototype implementation of our proposed solution was developed and tested on a testbed, demonstrating the feasibility of the solution.

The study also compares the performance of our proposed solution against the current approach based on re-establishing new IPsec tunnels, and the alternative approach of running IPsec over Mobile IP. Out of the three solutions, MIPsec has the lowest handoff latency. However, it also has the highest setup and teardown latency, as well as the highest bandwidth overhead. Our proposed solution has a similar bandwidth overhead when compared with that of standard IPsec. In terms of handoff latencies, our proposed solution has a lower handoff delay. The setup and teardown latency, however, are higher than that of standard IPsec.

Our proposed solution is no less secure than running IPsec over Mobile IP. Due to the use of Mobile IP registration messages, these solutions may be vulnerable to tunnel redirection attacks should an attacker somehow manages to compromise the Mobile IP authentication mechanism. In this situation, the encrypted data will remain secure since the attacker does not have access to the encryption keys.

# Appendix A

## Wireless Access Point Configuration

The Lucent WavePOINT-II configuration files for the IPSEC-HN home network and the IPSEC-FN1 foreign network are shown below. A similar configuration is used on the Lucent Orinoco AP-2000 for the IPSEC-FN2 foreign network. All the wireless access points are configured to operate in IEEE 802.11b mode with all security services, including WEP encryption, disabled.

### A.1 IPSEC-HN Configuration (IPSEC-HN.cnf)

```
# Configuration: IPSEC-HN.cnf 17336 bytes

hostname AP-HOME

bridging dhcp-response from-IP 1.1.1.1

ip address 10.0.0.254 255.0.0.0
ip default-gateway 10.0.0.1
ip ttl 64

access-list ethernet protocol permit others
access-list ethernet address permit-listed-only
access-list ap 0060.1D23.A766

# Ethernet
interface 1
  remote
  speed 10
  duplex half
done

# 802.11b
interface 2
  local
  multicast transmit-rate 2-mbps
```

```
encapsulation 802.11 access-point open IPSEC-HN
distance-between-aps small
no medium-reservation
dtim-period 1
done
```

## A.2 IPSEC-FN1 Configuration (IPSEC-FN1.cnf)

```
# Configuration: IPSEC-FN1.cnf 17336 bytes

hostname AP-FOREIGN1

bridging dhcp-response from-IP 1.1.1.1

ip address 30.0.0.254 255.0.0.0
ip default-gateway 30.0.0.2
ip ttl 64

access-list ethernet protocol permit others
access-list ethernet address permit-listed-only
access-list ap 0060.1D23.A766

# Ethernet
interface 1
  remote
  speed 10
  duplex half
done

# 802.11b
interface 2
  local
  multicast transmit-rate 2-mbps
  encapsulation 802.11 access-point open IPSEC-FN1
  distance-between-aps small
  no medium-reservation
  dtim-period 1
done
```

# Appendix B

## Dynamics Mobile IP Configuration

The Dynamics Mobile IP home agent and mobile node daemon configuration files are listed below. The configuration does not make use of any foreign agents. Instead, the mobile node is configured to operate in co-located care-of-address mode. Additionally, reverse tunnelling is employed for packets sent from the mobile node.

### B.1 Dynamics Home Agent Daemon Configuration (dynhad.conf)

```
# Home Agent configuration file

# Interfaces to be used for Mobile IP services.
# ha_disc:
#   1 = allow dynamic HA discovery with broadcast messages
# agentadv:
#   0 = do not send agent advertisements without agent solicitation
#   1 = send agent advertisements regularly
# interval: number of seconds to wait between two agentadvs

INTERFACES_BEGIN
# interface  ha_disc  agentadv  interval
eth0         1        1         10
eth1         1        0         20
INTERFACES_END

# UDP port to listen for registration requests.
UDPPort 434

# MaxBindings can be used to restrict the maximum number of Mobile Nodes
# that are concurrently attached to this Home Agent.
MaxBindings 20
```



```

# The default tunnel lifetime.
HADefaultTunnelLifetime 600

# The Registration error reply interval.
RegErrorReplyInterval 1

# Triangle tunnel means that the packages to MNs are send via the HA, but
# packages from MN are routed directly.
EnableTriangleTunneling TRUE

# Reverse tunnel means bi-directional tunneling in which both the packages
# from and to MN are send via HA.
EnableReverseTunneling TRUE

# The Home Agent needs to know what kind of security parameters each
# authorized Mobile Node uses.

AUTHORIZEDLIST_BEGIN
# SPI          IP
1000          10.0.0.11
AUTHORIZEDLIST_END

# The Home Agents needs a security association for each authorized Mobile
# Node. The association includes following information.
# SPI (Security Parameter Index): a key for the other fields.
# Authentication Algorithm:
#   4: HMAC-MD5 [RFC 2104]
# Replay Protection Method:
#   1: timestamps
# Timestamp tolerance indicates how many seconds the MN's timestamp can differ
# from the HA's clock.
# The maximum lifetime for the binding is given in seconds.
# Shared Secret: a secret data known by MN and HA.

SECURITY_BEGIN
#      auth.   replay  timestamp      max      shared
# SPI  alg.    meth.   tolerance    lifetime    secret
1000  4       1       120          600        "mobileipsec"
SECURITY_END

# Home Agent may have optional security associations with Foreign Agents.

FA_SECURITY_BEGIN

FA_SECURITY_END

# The Highest FA public key can be protected from man-in-the-middle style
# attacks between the HFA and the HA with hash code.
# If the hash code is received, check the public key with it.
PublicKeyHashMethod 1

```

```
# The log messages are written through syslog service.
SyslogFacility LOG_DAEMON
```

```
# API interfaces.
HAAPIReadSocketPath "/var/run/dynamics_ha_read"
HAAPIReadSocketGroup "root"
HAAPIReadSocketOwner "root"
HAAPIReadSocketPermissions 0766

HAAPIAdminSocketPath "/var/run/dynamics_ha_admin"
HAAPIAdminSocketGroup "root"
HAAPIAdminSocketOwner "root"
HAAPIAdminSocketPermissions 0700
```

```
END
```

## B.2 Dynamics Mobile Node Daemon Configuration (dynmnd.conf)

```
# Mobile Node configuration file

# The Mobile Nodes's IP address in the Home Network.
MNHomeIPAddress 10.0.0.11

# Disables AAA extensions.
UseAAA FALSE

# The IP address of Mobile Node's Home Agent.
HAIPAddress 20.0.0.1

# To allows MN to detect other HA's interfaces, their IP
# addresses may be configured here. MN will use this list in addition to
# HAIPAddress when determining whether an agent advertisement is from its own
# HA (i.e., when MN is at home).
AlternativeHAIPAddress 10.0.0.1

# Disallows AAA from assigning a home agent and home address from the
# foreign network.
AllowHomeAddrFromForeignNet FALSE

# Disable FA Decapsulation. Sets the default mode where the MN decapsulates
# the IP-within-IP encapsulated IP packets.
EnableFADecapsulation FALSE

# Network address of home network.
HomeNetPrefix 10.0.0.0/8
```

```

# The SPI (Security Parameter Index) is used for indexing the security
# association at the Home Agent.
SPI 1000

# The shared secret is used with the HA.
SharedSecret "mobileipsec"

# Authentication algorithm: HMAC-MD5 [RFC 2104]
AuthenticationAlgorithm 4

# Replay prevention method: time stamps
ReplayMethod 1

# Mobile Node does not have optional security associations with Foreign Agents.

FA_SECURITY_BEGIN

FA_SECURITY_END

# Tunneling Mode: accept only reverse tunnel
TunnelingMode 3

# When MN can get its own co-located care-of address and use reverse tunneling,
# the normal method is to set the default route to the tunnel. This means that
# all the packets destined to other networks than the current subnet in the
# visited network are send via the HA. If the co-located COA is public, it can
# be used for sessions that do not need constant IP address.
# MN Descapsulation Route Handling: set default route to the tunnel
MNDecapsRouteHandling 0

# DefaultTunnelLifetime is the lifetime suggested in registration.
# The request timer will be set according to this value.
MNDefaultTunnelLifetime 300

# UDP port to be used for sending registration requests.
UDPPort 434

# The log messages are written through syslog service.
SyslogFacility LOG_DAEMON

# Ignore these interfaces. No agent advertisements are received nor
# agent solicitations sent for these interfaces.

IGNORE_INTERFACES_BEGIN
lo
dummy0
tunl0
gre0
IGNORE_INTERFACES_END

```

```
# Other programs may set routing entries so that the data connection may
# fail. The MN can try to enforce the routes that it believes should be used.
EnforceRoutes FALSE

# Disable AP polling.
APPollingInterval -1

# Disable periodic agent solicitation.
SolicitationInterval -1

# API interfaces.
MNAPIReadSocketPath "/var/run/dynamics_mn_read"
MNAPIReadSocketGroup "root"
MNAPIReadSocketOwner "root"
MNAPIReadSocketPermissions 0666

MNAPIAdminSocketPath "/var/run/dynamics_mn_admin"
MNAPIAdminSocketGroup "root"
MNAPIAdminSocketOwner "root"
MNAPIAdminSocketPermissions 0700

END
```

# Appendix C

## Racoon Configuration

The Racoon IKE daemon configuration files, `racoon.conf`, for the home agent and the mobile node are included below. Currently, both the home agent and mobile node are configured to use pre-shared keys for the phase 1 remote peer authentication. The shared key should be defined in `/etc/racoon/psk.txt`.

### C.1 Home Agent Racoon Configuration

```
# Racoon IKE daemon configuration file.

path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";

remote anonymous {
    exchange_mode main;
    passive on;
    generate_policy on;
    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key;
        dh_group modp1024;
    }
}

sainfo anonymous {
    pfs_group 2;
    lifetime time 24 hours;
    encryption_algorithm 3des, blowfish 448, rijndael;
    authentication_algorithm hmac_shal, hmac_md5;
    compression_algorithm deflate;
}
```

## C.2 Mobile Node Racoon Configuration

```
# Racoon IKE daemon configuration file.

path include "/etc/racoon";
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";

remote 20.0.0.1 {
    exchange_mode main;
    proposal {
        encryption_algorithm 3des;
        hash_algorithm md5;
        authentication_method pre_shared_key;
        dh_group modp1024;
    }
}

sainfo anonymous {
    pfs_group 2;
    lifetime time 24 hours;
    encryption_algorithm 3des, blowfish 448, rijndael;
    authentication_algorithm hmac_sha1, hmac_md5;
    compression_algorithm deflate;
}
```

# Appendix D

## Basic Network Configuration Scripts

These shell scripts are used to configure the basic networking functionalities on the home agent, the foreign router, and the mobile node. They are also used to restore the systems to a consistent starting point for all the experiments.

### D.1 Home Agent Setup Script (ha-setup)

```
#!/bin/bash

#-----
# Home Agent Setup Script
#-----

# Turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

# Kill any home agent daemon and racoon daemon that are currently running
kill 'ps -e | grep dynha | awk '{print $1}''
kill 'ps -e | grep racoon | awk '{print $1}''

rm -f /tmp/.racoon

# Flush the SAD, SPD, and routing tables
setkey -F
setkey -FP

ifconfig lo down
ifconfig eth0 down
ifconfig eth1 down
ifconfig sit0 down
ifconfig tunl0 down
ifconfig TUNL0 down

ip route flush cache
```

```
# Bring up the interfaces and configure the IP routing table
ifconfig lo up
ifconfig eth0 10.0.0.1 up
ifconfig eth1 20.0.0.1 up

route add default gateway 20.0.0.2
```

## D.2 Foreign Router Setup Script (fa-setup)

```
#!/bin/bash

#-----
# Foreign Router Setup Script
#-----

# Turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

# Flush the routing tables
ifconfig lo down
ifconfig eth0 down
ifconfig eth1 down
ifconfig eth2 down
ifconfig sit0 down
ifconfig tunl0 down

ip route flush cache

# Bring up the interfaces and configure the IP routing table
ifconfig lo up
ifconfig eth0 20.0.0.2 up
ifconfig eth1 30.0.0.2 up
ifconfig eth2 40.0.0.2 up

route add -net 10.0.0.0/8 gateway 20.0.0.1
```

## D.3 Mobile Node Setup Script (mn-setup)

```
#!/bin/bash

#-----
# Mobile Node Setup Script
#-----
```



```

# Usage: mn-setup <ssid> <ip_address> <gateway>

DEF_IF=eth0

ESSID=$1
IP_ADDR=$2
GATEWAY=$3

# Turn on IP forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

# Kill any mobile node daemon and racoon daemon that are currently running
kill `ps -e | grep dynmn | awk '{print $1}'`
kill `ps -e | grep racoon | awk '{print $1}'`

rm -f /tmp/.racoon

# Flush the SAD, SPD, and routing tables
setkey -F
setkey -FP

ifconfig lo down
ifconfig eth0 down
ifconfig eth1 down
ifconfig sit0 down
ifconfig tunl0 down
ifconfig tunl1 down
ifconfig TUNLMNA down

ip tunnel del tunl1

ip rule del from 0.0.0.0
ip rule del to 20.0.0.1

ip route flush table 200
ip route flush cache

iptables -F OUTPUT
iptables -t nat -F POSTROUTING

# Bring up the interfaces
ifconfig lo up

iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $IP_ADDR up

route add default gateway $GATEWAY

```

# Appendix E

## Standard IPsec Scripts

Standard IPsec refers to the current approach of re-establishing new IPsec tunnels as the mobile node moves between different networks. The implementation is based on the road warrior configuration, as suggested in the IPsec HOWTO [18]. The scripts used on the home agent and the mobile node are listed below.

### E.1 Home Agent IPsec Script (ha-ipsec)

```
#!/bin/bash

#-----
# Home Agent IPsec Script
#-----

# Flush the SAD and SPD
setkey -F;
setkey -FP;

racoon
```

### E.2 Mobile Node IPsec Setup Script (mn-ipsec-setup)

```
#!/bin/bash

#-----
# Mobile Node IPsec Setup Script
#-----

# Usage: mn-ipsec-setup <essid> <ip_addr> <gateway>

DEF_IF=eth0
```

```

HA_EXT_IP=20.0.0.1

ESSID=$1
MN_COA_IP=$2
GATEWAY=$3

ifconfig $DEF_IF down
iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $MN_COA_IP up

route add default gateway $GATEWAY

# Start the racoon daemon
racoon

# IPsec security policies
echo "\
spdadd $MN_COA_IP 0.0.0.0/0 any -P out ipsec
    esp/tunnel/$MN_COA_IP-$HA_EXT_IP/require;
spdadd 0.0.0.0/0 $MN_COA_IP any -P in ipsec
    esp/tunnel/$HA_EXT_IP-$MN_COA_IP/require;
" | setkey -c

```

## E.3 Mobile Node IPsec Handoff Script (mn-ipsec-handoff)

```

#!/bin/bash

#-----
# Mobile Node IPsec Handoff Script
#-----

# Usage: mn-ipsec-handoff <ssid> <ip_addr> <gateway>

DEF_IF=eth0

HA_EXT_IP=20.0.0.1

ESSID=$1
MN_COA_IP=$2
GATEWAY=$3

ifconfig $DEF_IF down
iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $MN_COA_IP up

route add default gateway $GATEWAY

```

```

iptables -A OUTPUT -p icmp -j DROP

# IPsec security policies
setkey -F
setkey -FP

echo "\
spdadd 0.0.0.0/0 0.0.0.0/0 any -P out ipsec
      esp/tunnel/$MN_COA_IP-$HA_EXT_IP/require;
spdadd 0.0.0.0/0 0.0.0.0/0 any -P in ipsec
      esp/tunnel/$HA_EXT_IP-$MN_COA_IP/require;
" | setkey -c

iptables -F OUTPUT

```

## E.4 Mobile Node IPsec Teardown Script (mn-ipsec-teardown)

```

#!/bin/bash

#-----
# Mobile Node IPsec Teardown Script
#-----

# Usage: mn-ipsec-teardown <essid> <ip_addr> <gateway>

DEF_IF=eth0

ESSID=$1
MN_COA_IP=$2
GATEWAY=$3

kill `ps -e | grep racoon | awk '{print $1}'`

setkey -F
setkey -FP

ifconfig $DEF_IF down
iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $MN_COA_IP up

route add default gateway $GATEWAY

```

# Appendix F

## MIPsec Scripts

MIPsec refers to our implementation of IPsec over Mobile IP. The shell scripts for the home agent and the mobile node are included below.

### F.1 Home Agent MIPsec Setup Script (ha-mipsec)

```
#!/bin/bash

#-----
# Home Agent MIPsec Script
#-----

HA_EXT_IP=20.0.0.1
MN_HOM_IP=10.0.0.11

# Disables Path MTU discovery
# PMTU discovery does not function correctly when IPsec and Mobile IP tunnelling
# are used concurrently
echo 1 > /proc/sys/net/ipv4/ip_no_pmtu_disc

# Start the home agent daemon
dynhad

DYNHA_CMD='dynha_tool -p /var/run/dynamics_ha_admin'

# Returns the mobile node COA in $MN_CUR_IP
function getCurrentCOA {
    MN_CUR_IP='$DYNHA_CMD show $MN_HOM_IP $HA_EXT_IP | grep 'care-of' \
        | awk '{print $3}''
}

while true
do
```

```

# Wait for the mobile node to connect from a foreign network
echo 'Waiting for mobile node to connect...'
getCurrentCOA

while [[ $MN_CUR_IP == $MN_HOM_IP || $MN_CUR_IP == '' ]]
do
    getCurrentCOA
done

MN_COA_IP=$MN_CUR_IP
echo "Mobile node COA: $MN_COA_IP"

# Start the racoon daemon
racoon

# Security policies
echo "\
spdadd $HA_EXT_IP[434] 0.0.0.0/0[any] any -P out none;
spdadd 0.0.0.0/0[any] $HA_EXT_IP[434] any -P in none;
spdadd $HA_EXT_IP $MN_HOM_IP any -P out ipsec
    esp/tunnel/$HA_EXT_IP-$MN_HOM_IP/require;
spdadd $MN_HOM_IP $HA_EXT_IP any -P in ipsec
    esp/tunnel/$MN_HOM_IP-$HA_EXT_IP/require;
" | setkey -c

echo 'Home Agent setup completed.'

# Check for mobile COA change
while true
do
    getCurrentCOA

    if [[ $MN_COA_IP != $MN_CUR_IP ]]
    then
        # Teardown IPsec tunnels if mobile node is in home network
        if [[ $MN_CUR_IP == '' ]]
        then
            setkey -F
            setkey -FP
            echo 'Mobile Node at home.'
            break
        fi

        MN_COA_IP=$MN_CUR_IP
    fi

    sleep 1
done
done

```

## F.2 Mobile Node MIPsec Setup Script (mn-mipsec-setup)

```
#!/bin/bash

#-----
# Mobile Node MIPsec Setup Script
#-----

# Usage: mn-mipsec-setup <essid> <ip_addr> <gateway>

DEF_IF=eth0

HA_EXT_IP=20.0.0.1
MN_HOM_IP=10.0.0.11

ESSID=$1
MN_COA_IP=$2
GATEWAY=$3

DYNMN_CMD='dynmn_tool -p /var/run/dynamics_mn_admin'

# Advanced routing table ID
AR_ID=200
AR_TUN=tun11

ifconfig $DEF_IF down
iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $MN_COA_IP up

route add default gateway $GATEWAY

# Start the mobile node daemon (in the disconnected state)
dynmnd --disconnect
$DYNMN_CMD con HA

# Configure advanced routing
# Force the mobile node to use $MN_HOM_IP as the source address
ip tunnel add $AR_TUN local $MN_COA_IP remote $HA_EXT_IP dev $DEF_IF mode ipip
ifconfig $AR_TUN $MN_HOM_IP up
ip rule add to $HA_EXT_IP table $AR_ID
ip route add default dev $AR_TUN table $AR_ID

# Start the racoon daemon
racoon

# IPsec security policies
echo "\
spdadd 0.0.0.0/0[any] $HA_EXT_IP[434] any -P out none;
```

```

spdadd $HA_EXT_IP[434] 0.0.0.0/0[any] any -P in none;
spdadd $MN_HOM_IP 0.0.0.0/0 any -P out ipsec
    esp/tunnel/$MN_HOM_IP-$HA_EXT_IP/require;
spdadd 0.0.0.0/0 $MN_HOM_IP any -P in ipsec
    esp/tunnel/$HA_EXT_IP-$MN_HOM_IP/require;
" | setkey -c

# echo 'Mobile Node setup completed.'
```

## F.3 Mobile Node MIPsec Handoff Script (mn-mipsec-handoff)

```

#!/bin/bash

#-----
# Mobile Node MIPsec Handoff Script
#-----

# Usage: mn-mipsec-handoff <essid> <ip_addr> <gateway>

DEF_IF=eth0

HA_EXT_IP=20.0.0.1
MN_HOM_IP=10.0.0.11

ESSID=$1
MN_COA_IP=$2
GATEWAY=$3

DYNMN_CMD='dynmn_tool -p /var/run/dynamics_mn_admin'

# Advanced routing table ID
AR_ID=200
AR_TUN=tunl1

MN_OLD_IP='ifconfig $DEF_IF | grep 'inet ' \
    | awk '{print $2}' | awk -F : '{print $2}''

ifconfig $DEF_IF down
iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $MN_COA_IP up

route add default gateway $GATEWAY

# Install blocking route to prevent packets from 'leaking' unencrypted
iptables -A OUTPUT -s $MN_OLD_IP -p udp --destination-port ! 434 -j DROP
iptables -A OUTPUT -p icmp -j DROP
```



```

# Remove advanced route
ip route flush table $AR_ID

# Update the mobile IP tunnel
$DYNMN_CMD update $DEFAULT_IF
$DYNMN_CMD disconnect
$DYNMN_CMD connect HA

# Reinstall advanced route and remove blocking route
ip tunnel change $AR_TUN local $MN_COA_IP
ip route add default dev $AR_TUN table $AR_ID
iptables -F OUTPUT

```

## F.4 Mobile Node MIPsec Teardown Script (mn-mipsec-teardown)

```

#!/bin/bash

#-----
# Mobile Node MIPsec Teardown Script
#-----

# Usage: mn-mipsec-teardown <essid> <ip_addr> <gateway>

DEF_IF=eth0

HA_EXT_IP=20.0.0.1
MN_HOM_IP=10.0.0.11

ESSID=$1
MN_COA_IP=$2
GATEWAY=$3

DYNMN_CMD='dynamn_tool -p /var/run/dynamics_mn_admin'

# Advanced routing table ID
AR_ID=200
AR_TUN=tunl1

ifconfig $DEF_IF down
iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $MN_COA_IP up

route add default gateway $GATEWAY

kill `ps -e | grep racoon | awk '{print $1}'`

```

```
ip rule del to $HA_EXT_IP
ip tunnel del tunl1

# Update the mobile IP tunnel
$DYNMN_CMD update $MN_COA_IP

setkey -F
setkey -FP
```

# Appendix G

## DynIPsec Scripts

DynIPsec refers to our proposed solution of using Mobile IP registration messages for dynamic updates of the IPsec tunnel endpoints. The shell script implementation of the solution is detailed below.

### G.1 Home Agent DynIPsec Script (ha-dynipsec)

```
#!/bin/bash

#-----
# Home Agent DynIPsec Script
#-----

HA_EXT_IP=20.0.0.1
MN_HOM_IP=10.0.0.11

# Start the home agent daemon
dynhad

DYNHA_CMD='dynha_tool -p /var/run/dynamics_ha_admin'

# Returns the mobile node COA in $MN_CUR_IP
function getCurrentCOA {
    MN_CUR_IP='$DYNHA_CMD show $MN_HOM_IP $HA_EXT_IP | grep 'care-of' \
        | awk '{print $3}''
}

while true
do
    # Wait for the mobile node to connect from a foreign network

    echo 'Waiting for mobile node to connect...'
    getCurrentCOA
```

```

while [[ $MN_CUR_IP == $MN_HOM_IP || $MN_CUR_IP == '' ]]
do
    getCurrentCOA
done

MN_COA_IP=$MN_CUR_IP
echo "Mobile node COA: $MN_COA_IP"

# Start the racoon daemon
racoon

# Security policies
echo "\
spdflush;
spdadd $HA_EXT_IP[434] 0.0.0.0/0[any] any -P out none;
spdadd 0.0.0.0/0[any] $HA_EXT_IP[434] any -P in none;
spdadd $HA_EXT_IP $MN_HOM_IP any -P out ipsec
    esp/tunnel/$HA_EXT_IP-$MN_COA_IP/require;
spdadd $MN_HOM_IP $HA_EXT_IP any -P in ipsec
    esp/tunnel/$MN_COA_IP-$HA_EXT_IP/require;
" | setkey -c

echo 'Home Agent setup completed.'

# Check for mobile COA change
while true
do
    getCurrentCOA

    if [[ $MN_COA_IP != $MN_CUR_IP ]]
    then
        kill -9 `ps -e | grep racoon | awk '{print $1}'` > /dev/null 2>&1
        rm -f /tmp/.racoon

        # Teardown IPsec tunnels if mobile node is in home network
        if [[ $MN_CUR_IP == '' ]]
        then
            setkey -F
            setkey -FP
            break
        fi

        updaddr $MN_COA_IP $MN_CUR_IP
        echo "Updated SAD/SPD: $MN_COA_IP->$MN_CUR_IP"

        MN_COA_IP=$MN_CUR_IP
    fi

    sleep 1
done
done

```

## G.2 Mobile Node DynIPsec Setup Script (mn-dynipsec-setup)

```
#!/bin/bash

#-----
# Mobile Node DynIPsec Setup Script
#-----

# Usage: mn-dynipsec-setup <essid> <ip_addr> <gateway>

DEF_IF=eth0

HA_EXT_IP=20.0.0.1
MN_HOM_IP=10.0.0.11

ESSID=$1
MN_COA_IP=$2
GATEWAY=$3

DYNMN_CMD='dynmn_tool -p /var/run/dynamics_mn_admin'

# Advanced routing table ID
AR_ID=200

ifconfig $DEF_IF down
iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $MN_COA_IP up

route add default gateway $GATEWAY

# Start the mobile node daemon (in the disconnected state)
dynmnd --disconnect
$DYNMN_CMD con HA

# Start the racoon daemon
racoon

# IPsec security policies
echo "\
spdf flush;
spdadd 0.0.0.0/0[any] $HA_EXT_IP[434] any -P out none;
spdadd $HA_EXT_IP[434] 0.0.0.0/0[any] any -P in none;
spdadd 0.0.0.0/0 0.0.0.0/0 any -P out ipsec
    esp/tunnel/$MN_COA_IP-$HA_EXT_IP/require;
```

```

spdadd 0.0.0.0/0 0.0.0.0/0 any -P in ipsec
    esp/tunnel/$HA_EXT_IP-$MN_COA_IP/require;
" | setkey -c

# Configure advanced routing
# Force the mobile node to use $MN_HOM_IP as the source address
ip rule add from 0.0.0.0 table $AR_ID
ip route add default dev $DEF_IF src $MN_HOM_IP via $GATEWAY table $AR_ID

iptables -t nat -A POSTROUTING -s $MN_HOM_IP -j SNAT --to-source $MN_COA_IP

# echo 'Mobile Node setup completed.'

```

## G.3 Mobile Node DynIPsec Handoff Script (mn-dynipsec-handoff)

```

#!/bin/bash

#-----
# Mobile Node DynIPsec Handoff Script
#-----

# Usage: mn-dynipsec-handoff <ssid> <ip_addr> <gateway>

DEF_IF=eth0

HA_EXT_IP=20.0.0.1
MN_HOM_IP=10.0.0.11

ESSID=$1
MN_COA_IP=$2
GATEWAY=$3

DYNMN_CMD='dynmn_tool -p /var/run/dynamics_mn_admin'

# Advanced routing table ID
AR_ID=200

MN_OLD_IP='ifconfig $DEF_IF | grep 'inet ' \
    | awk '{print $2}' | awk -F : '{print $2}''

ifconfig $DEF_IF down
iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $MN_COA_IP up

route add default gateway $GATEWAY

```

```

# Forces the mobile node to use the new COA
iptables -t nat -F POSTROUTING
iptables -t nat -A POSTROUTING -s $MN_HOM_IP -j SNAT --to-source $MN_COA_IP
ip route flush cache

# Update the mobile IP tunnel
$DYNMN_CMD update $DEF_IF
$DYNMN_CMD disc
$DYNMN_CMD con HA

kill -9 `ps -e | grep racoon | awk '{print $1}'` > /dev/null 2>&1
rm -f /tmp/.racoon

# Update the SAD/SPD
updatadr $MN_OLD_IP $MN_COA_IP

# Reinstall advanced route
ip route add default dev $DEF_IF src $MN_HOM_IP via $GATEWAY table $AR_ID

```

## G.4 Mobile Node DynIPsec Teardown Script (mn-dynipsec-teardown)

```

#!/bin/bash

#-----
# Mobile Node DynIPsec Teardown Script
#-----

# Usage: mn-dynipsec-teardown <ssid> <ip_addr> <gateway>

DEF_IF=eth0

HA_EXT_IP=20.0.0.1
MN_HOM_IP=10.0.0.11

ESSID=$1
MN_COA_IP=$2
GATEWAY=$3

DYNMN_CMD='dynmn_tool -p /var/run/dynamics_mn_admin'

ifconfig $DEF_IF down
iwconfig $DEF_IF essid $ESSID
ifconfig $DEF_IF $MN_COA_IP up

route add default gateway $GATEWAY

```

```
ip rule del from 0.0.0.0
iptables -t nat -F POSTROUTING

# Update the mobile IP tunnel
$DYNMN_CMD update $MN_COA_IP

kill `ps -e | grep racoon | awk '{print $1}'`

setkey -F
setkey -FP
```



# Appendix H

## Update Address

The `updaddr` program is used to “update” the IP address parameters in the SAD and SPD at the home agent and the mobile node. Usage:

```
updaddr <old_addr> <new_addr>
```

The program replaces all IP address parameters in the SAD and SPD that match `<old_addr>` with `<new_addr>`.

### H.1 Source Code (`updaddr.c`)

```
#include <assert.h>
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXLEN_NAME 10
#define MAXLEN_LINE 1024
#define MAXLEN_CMD 1024

FILE *infile, *outfile;

void updateSAD(char *oldAddr, char *newAddr) {
    char input[MAXLEN_LINE];
    char srcAddr[16], dstAddr[16];
    char proto[4], mode[10];
    char authAlg[15], authKey[513];
    char encAlg[15], encKey[513];
    int spi, lh, ls, bh, bs;

    char *token, *alg, *ptr, *idx;
    int authRequired, encRequired;

    while(fgets(input, MAXLEN_LINE, infile)) {
```

```

/* Scan for the start of SAD entry */
if(isdigit(input[0])) {
    authRequired = 0;
    encRequired = 0;

    sscanf(input, "%s %s", srcAddr, dstAddr);

    fgets(input, MAXLEN_LINE, infile);
    sscanf(input, " %s mode=%s spi=%d(", proto, mode, &spi);

    /* Parse the authentication and encryption keys */
    while(fgets(input, MAXLEN_LINE, infile)) {
        if(input[1] == 'A') {
            authRequired = 1;
            alg = authAlg;
            ptr = authKey;
        }
        else if(input[1] == 'E') {
            encRequired = 1;
            alg = encAlg;
            ptr = encKey;
        }
        else {
            break;
        }

        /* Remove the trailing newline character */
        if(idx = strchr(input, '\n'))
            *idx = '\0';

        token = strtok(input + 3, " ");
        strcpy(alg, token);

        while(token = strtok(NULL, " ")) {
            strcpy(ptr, token);
            ptr += strlen(token);
        }
    }

    /* Parse the lh, ls, bh, bs values */
    fgets(input, MAXLEN_LINE, infile);
    fgets(input, MAXLEN_LINE, infile);
    sscanf(input, " diff: %s hard: %d(s) soft: %d(", &lh, &ls);
    fgets(input, MAXLEN_LINE, infile);
    fgets(input, MAXLEN_LINE, infile);
    sscanf(input, " current: %s hard: %d(bytes) soft: %d(", &bh, &bs);

    /* Output the updated SAD entry (if necessary) */
    if(!strcmp(srcAddr, oldAddr)) {
        fprintf(outfile, "delete %s %s %s %d;\n", srcAddr, dstAddr, proto, spi);
        strcpy(srcAddr, newAddr);
    }
}

```

```

    }
    else if(!strcmp(dstAddr, oldAddr)) {
        fprintf(outfile, "delete %s %s %s %d;\n", srcAddr, dstAddr, proto, spi);
        strcpy(dstAddr, newAddr);
    }
    else {
        continue;
    }

    fprintf(outfile, "add %s %s %s %d -m %s -lh %d -ls %d -bh %d -bs %d",
        srcAddr, dstAddr, proto, spi, mode, lh, ls, bh, bs);

    if(encRequired) fprintf(outfile, " -E %s 0x%s", encAlg, encKey);
    if(authRequired) fprintf(outfile, " -A %s 0x%s", authAlg, authKey);

    fprintf(outfile, ";\n");
}
}

void updateSPD(char *oldAddr, char *newAddr) {
    char input[MAXLEN_LINE];
    char cmd[MAXLEN_CMD];
    char iden[128], dir[4], policy[8];
    char *start, *end;
    int len;

    /* Scan for the start of SPD entry */
    while(fgets(input, MAXLEN_LINE, infile)) {
        if(isdigit(input[0])) {
            strcpy(iden, input);
            fscanf(infile, " %s %s \n", dir, policy);

            sprintf(cmd, "spdadd %s-P %s %s\n", iden, dir, policy);

            while(fgets(input, MAXLEN_LINE, infile)) {
                if(strstr(input, "esp") || strstr(input, "ah"))
                    strcat(cmd, input);
                else
                    break;
            }

            fprintf(outfile, "spdddelete %s-P %s %s;\n", iden, dir, policy);
            start = cmd;

            if(end = strstr(start, oldAddr)) {
                /* Output the updated SPD entry */
                while(end) {
                    *end = '\0';
                    fprintf(outfile, "%s%s", start, newAddr);
                    len = strlen(oldAddr);

```

```

        start = end + len;
        end = strstr(start, oldAddr);
    }

    fprintf(outfile, "%s;\n", start);
}
else {
    fprintf(outfile, "%s;\n", cmd);
}
}
}
}

int main(int argc, char *argv[]) {
    char infileName[MAXLEN_NAME] = "inXXXXXX";
    char outfileName[MAXLEN_NAME] = "outXXXXXX";
    char cmd[MAXLEN_CMD];
    int infileFd, outfileFd;

    /* Usage: argv[0] <old-ip-addr> <new-ip-addr> */
    assert(argc == 3);

    /* Create temp files to store the SAD/SPD dump and the setkey commands */
    infileFd = mkstemp(infileName);
    outfileFd = mkstemp(outfileName);

    if(infileFd == -1 || outfileFd == -1)
        return(1);

    infile = fdopen(infileFd, "r");
    outfile = fdopen(outfileFd, "w");

    if(!infile || !outfile)
        return(2);

    /* Dump the SAD */
    sprintf(cmd, "setkey -D > %s", infileName);
    system(cmd);

    updateSAD(argv[1], argv[2]);

    /* Dump the SPD */
    sprintf(cmd, "setkey -DP >> %s", infileName);
    system(cmd);

    updateSPD(argv[1], argv[2]);

    fclose(outfile);

    /* Update the SAD and SPD */
    sprintf(cmd, "setkey -f %s", outfileName);

```

```
system(cmd);

fclose(infile);

/* Remove the temporary files */
sprintf(cmd, "rm -f %s %s", inFileName, outFileName);
system(cmd);

return 0;
}
```

# Bibliography

- [1] *Dynamics Mobile IP*. <http://dynamics.sourceforge.net/>.
- [2] *IPsec-Tools*. <http://ipsec-tools.sourceforge.net/>.
- [3] *Racoon*. <http://www.kame.net/racoon/>.
- [4] K. Byoung-Jo and S. Srinivasan. Simple mobility support for IPsec tunnel mode. *Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th*, 3:1999–2003, 2003.
- [5] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. *RFC 2460*, Dec. 1998. <http://www.ietf.org/rfc/rfc2460.txt>.
- [6] M. Garrels. *Introduction to Linux - A Hands on Guide*. The Linux Documentation Project, Sep. 2005. <http://www.tldp.org/LDP/intro-linux/html/index.html>.
- [7] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). *RFC 2409*, Nov. 1998. <http://www.ietf.org/rfc/rfc2409.txt>.
- [8] The Internet Engineering Task Force. *IPsec*. <http://www.ietf.org/html.charters/ipsec-charter.html>.
- [9] The Internet Engineering Task Force. *Mobile IP*. <http://www.ietf.org/html.charters/mobileip-charter.html>.
- [10] D. Johnson, C. Perkins, and J. Arkko. Mobility Support for IPv6. *RFC 3775*, Jun. 2004. <http://www.ietf.org/rfc/rfc3775.txt>.
- [11] E. Kaufman and A. Newman. *Implementing IPsec: making security work on VPNs, intranets and extranets*. Wiley, 1999.
- [12] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. *RFC 2401*, Nov. 1998. <http://www.ietf.org/rfc/rfc2401.txt>.
- [13] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP). *RFC 2408*, Nov. 1998. <http://www.ietf.org/rfc/rfc2408.txt>.

- [14] J. Mogul and S. Deering. Path MTU Discovery. *RFC 1191*, Nov. 1990.  
<http://www.ietf.org/rfc/rfc1191.txt>.
- [15] National Laboratory for Applied Network Research. *NLANR/DAST: Iperf - The TCP/UDP Bandwidth Measurement Tool*. <http://dast.nlanr.net/Projects/Iperf/>.
- [16] C. Perkins. IP Mobility Support for IPv4. *RFC 3344*, Aug. 2002.  
<http://www.ietf.org/rfc/rfc3344.txt>.
- [17] Portland State University. *The Portland State University Secure Mobile Networking Project*. <http://www.cs.pdx.edu/research/SMN/>.
- [18] R. Spenneberg. *IPsec HOWTO*. <http://www.ipsec-howto.org/>.